Deep Generative Models

Lecture 18, 11/14/17

David Sontag

Massachusetts Institute of Technology

Acknowledgement: several slides adapted from Shakir Mohamed, Danilo Rezende, Rahul Krishnan, Rajesh Ranganath, Dave Blei, and Joan Bruna

Course Announcements

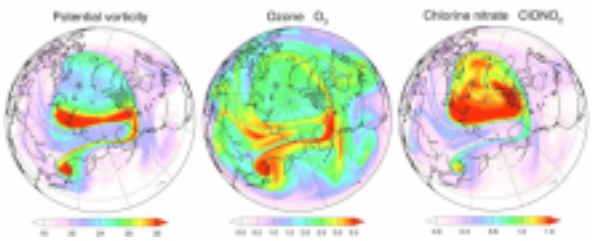
- HW3 due tonight
- Exercise 10 posted shortly (last one covering exam material)
- Reference for today's class:
  - *Deep Learning* (Section 20.9-20.11), Goodfellow, Bengio, Courville. MIT Press, 2016
  - Additional references at the end of lecture slides

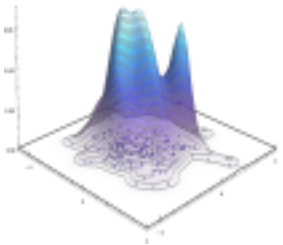
Recap: What is a Generative Model?

A model that allows us to learn a simulator of data	Models that allow for (conditional) density estimation	Approaches for unsupervised learning of data

Characteristics are:

- **Probabilistic** models of data that allow for uncertainty to be captured.
- **Data distribution p(x)** is targeted.
- **High-dimensiona**l outputs.

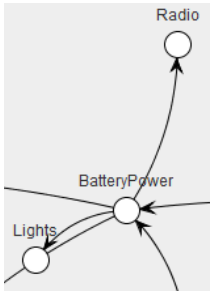


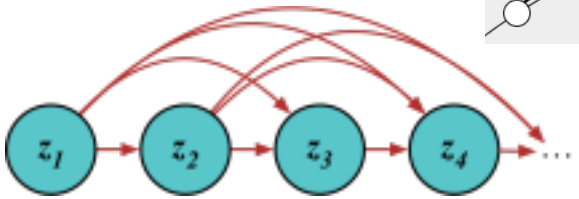


Types of Generative Models

Fully-observed models

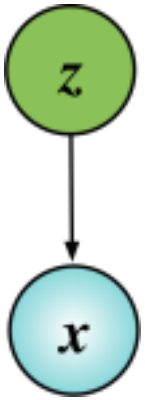
Model observed data directly without introducing any new unobserved local variables.

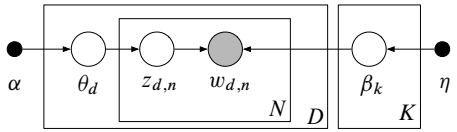




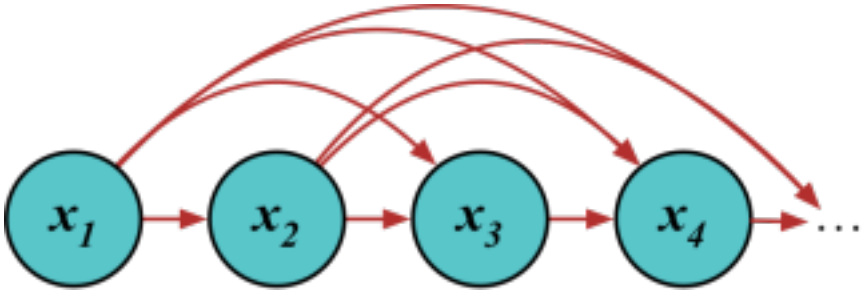
Latent Variable Models

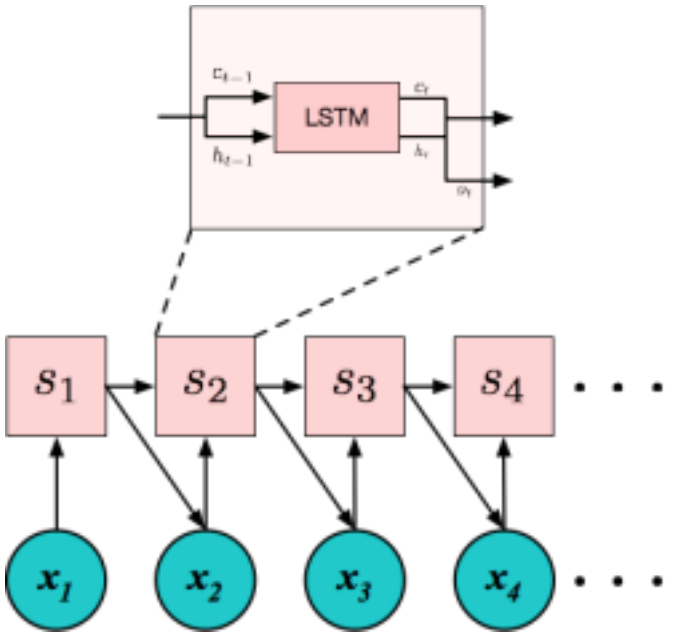
Introduce an unobserved random variable for every observed data point to explain hidden causes.





Building Generative Models





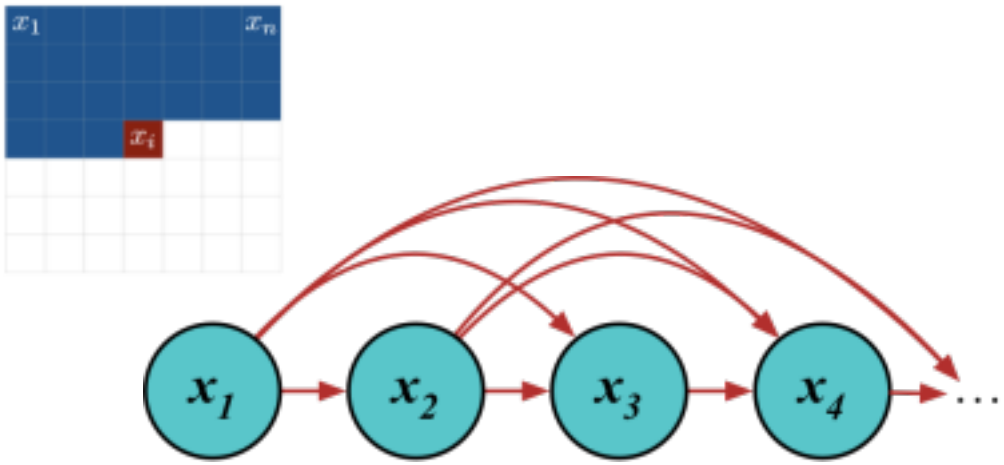
$$p(x_{1,\ldots,N}) = \prod_{i=1}^{N} p(x_i | x_{1,\ldots,(i-1)})$$

$$p(x_{1,\ldots,N}) = \prod_{i=1}^{N} p(x_i | s_i(s_{i-1}, x_{i-1}))$$

Equivalent ways of representing the same DAG

Fully-observed Models

$$p(x_{1,\ldots,N}) = \prod_{i=1}^{N} p(x_i | x_{1,\ldots,(i-1)})$$



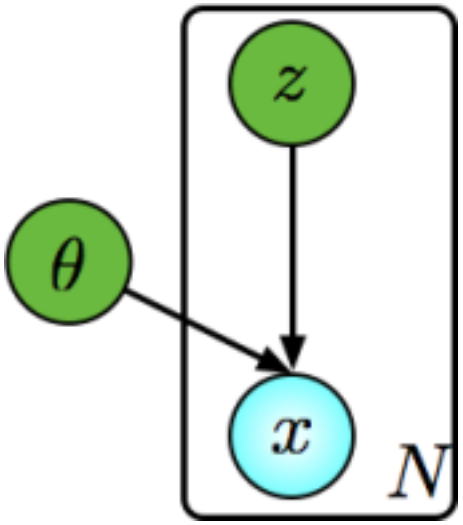
All conditional probabilities described by deep networks.

+ Can directly encode how observed points are related.
+ Any data type can be used
+ Directed graphical models (Bayesian networks):
  Parameter learning simple
+ Log-likelihood is directly computable, no approximation needed.
+ Easy to scale-up to large models, many optimisation tools available.

- Order sensitive.
- Not always clear how to parameterize and perform parameter sharing
- Generation can be slow: iterate through elements sequentially, or using a Markov chain.

Building Generative Models

Graphical Models + Computational Graphs (aka NNets)

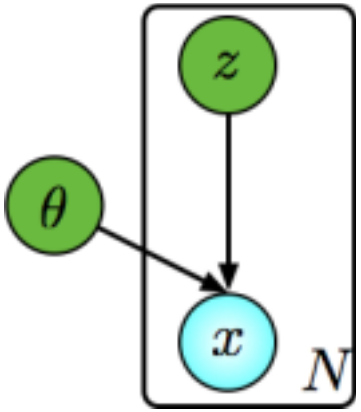


$$p(x, z, \theta) = \rho(\theta) \prod_{i=1}^{N} p(x_i|z_i, \theta)\pi(z_i)$$

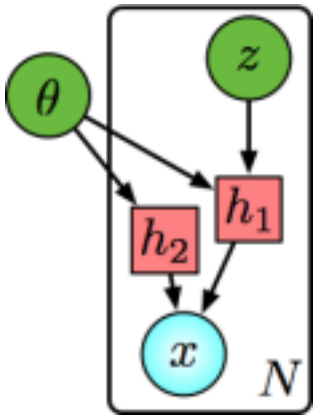
$$\begin{aligned}
\pi(z) &= \mathcal{N}(0, \mathbb{I}_{d_z}) \\
\rho(\theta) &= \mathcal{N}(0, \kappa^2 \mathbb{I}_{d_\theta}) \\
p(x|z, \theta) &= \mathcal{N}(\theta_0 + \theta_1 z, \exp(\theta_2)) \\
\theta &= \{\theta_0 \in \mathbb{R}^{d_x}, \theta_1 \in \mathbb{R}^{d_x \times d_z}, \theta_2 \in \mathbb{R}^{d_x}\}
\end{aligned}$$

Building Generative Models

Graphical Models + Computational Graphs (aka NNets)



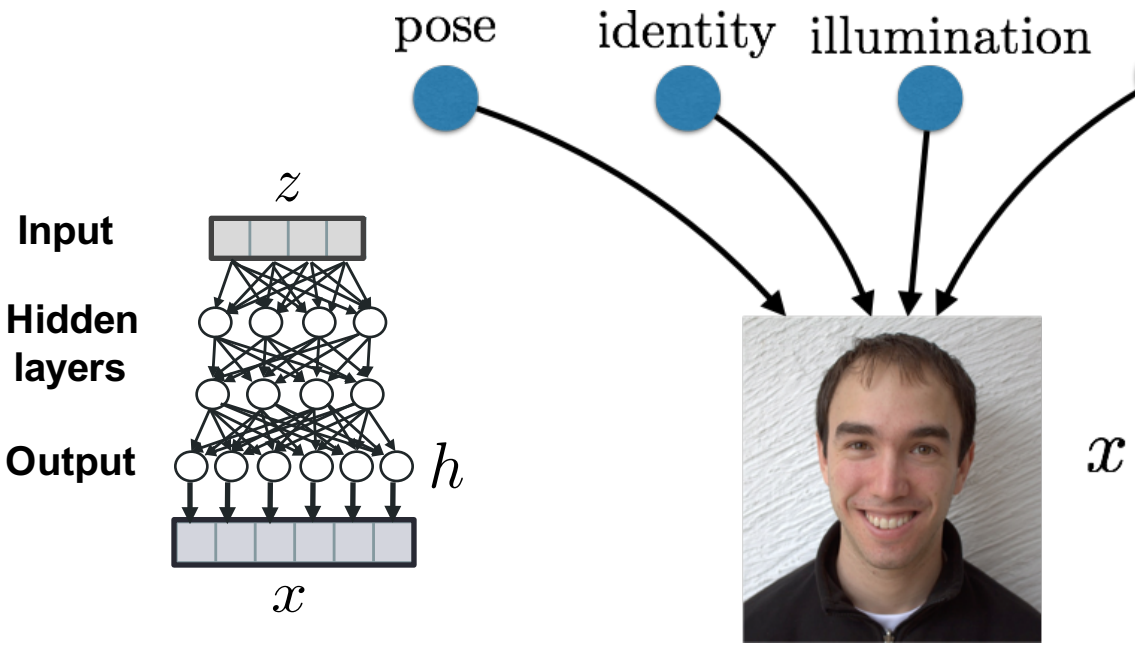
$$
\begin{aligned}
\pi(z) &= \mathcal{N}(0, \mathbb{I}_{d_z}) \\
\rho(\theta) &= \mathcal{N}(0, \kappa^2 \mathbb{I}_{d_\theta}) \\
p(x|z,\theta) &= \mathcal{N}(\theta_0 + \theta_1 z, \exp(\theta_2))
\end{aligned}
$$



$$
\begin{aligned}
\pi(z) &= \mathcal{N}(0, \mathbb{I}_{d_z}) \\
\rho(\theta) &= \mathcal{N}(0, \kappa^2 \mathbb{I}_{d_\theta}) \\
h_1 &= \theta_0 + \theta_1 z \\
h_2 &= \exp(\theta_2) \\
p(x|z,\theta) &= \mathcal{N}(h_1, h_2)
\end{aligned}
$$

Building Richer Generative Models

- Imagine we wanted to do factor analysis on much more complex data, such as images:



$$p(z) \sim \mathcal{N}(0, I)$$

$$h = f_{\text{neural net}}(z)$$

$$x_i \sim \text{Bernoulli}(h_i)$$

[Bernoulli because I am assuming black & white image]

How do we learn these *deep* generative models?

Today's lecture

1. Learning by maximizing the ELBO (Evidence Lower Bound)
2. Compiling (approximate) inference with recognition networks
3. Stochastic gradient estimators
4. Semi-supervised learning
5. Going beyond mean-field assumption

Background: Jensen's inequality

- Jensen's Inequality: For concave f, we have

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

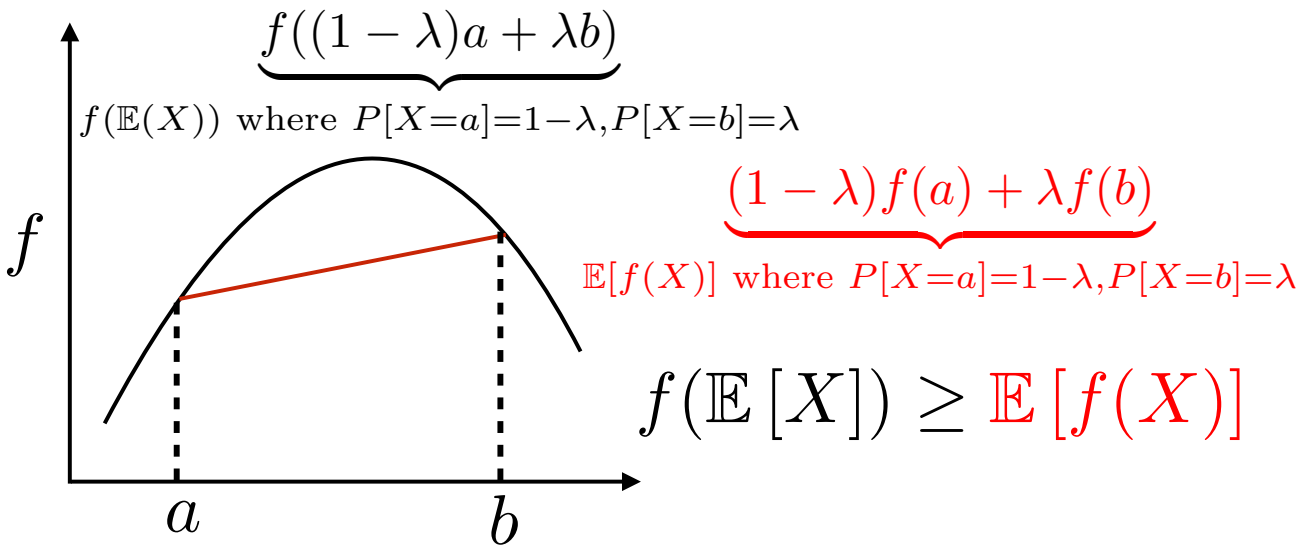
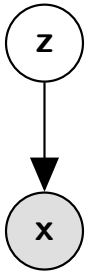


Figure: Jensen's Inequality

Background: Maximum likelihood estimation

- We assume that for $\mathcal{D} = \{x_1, \ldots, x_N\}$, $x_i \sim p(x)$ i.i.d
- We hypothesize a model (with parameters θ) for how the data is generated
- The Maximum Likelihood Principle: $\max_\theta p(\mathcal{D}; \theta) = \prod_{i=1}^N p(x_i; \theta)$
- Typically work with the log probability: i.e $\max_\theta \sum_{i=1}^N \log p(x_i; \theta)$

A simple Bayesian Network



- Lets start with a very simple generative model for our data
- We assume that the data is generated i.i.d as:

$$z \sim p(z) \qquad x \sim p(x|z)$$

- z is latent/hidden and x is observed

Bounding the Marginal Likelihood

- Log-Likelihood of a single datapoint $x \in \mathcal{D}$ under the model: $\log p(x; \theta)$
- Important: Assume $\exists q(z; \phi)$, (variational approximation)

$$\log p(x) = \log \int_z p(x, z) \text{ (Multiply and divide by } q(z))$$

$$= \log \int_z \frac{q(z)p(x, z)}{q(z)} = \log \mathbb{E}_{z \sim q(z)} \left[\frac{p(x, z)}{q(z)} \right] \text{ (By Jensen's Inequality)}$$

$$\geq \int_z q(z) \log \frac{p(x, z)}{q(z)} = \mathcal{L}(x; \theta, \phi)$$

$$= \underbrace{\mathbb{E}_{q(z;\phi)}[\log p(x, z; \theta)]}_{\text{Expectation of Joint distribution}} + \underbrace{\mathrm{H}(q(z; \phi))}_{\text{Entropy}}$$

Evidence Lower BOund (ELBO)/Variational Bound

- When is the lower bound tight?
- Look at: function - lower bound

$$\log p(x;\theta) - \mathcal{L}(x;\theta,\phi)$$

$$\begin{aligned}
&\log p(x) - \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \int_z q(z) \log p(x) - \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \int_z q(z) \log \frac{q(z)p(x)}{p(x,z)} \\
&= \int_z q(z) \log \frac{q(z)}{p(z|x)} = \mathrm{KL}(q(z;\phi)||p(z|x;\theta)).
\end{aligned}$$

Evidence Lower BOund (ELBO)/Variational Bound

- We assumed the existance of $q(z;\phi)$
- What we just showed is that:

Key Point

The optimal $q(z;\phi)$ corresponds to the one that realizes
$\mathrm{KL}(q(z;\phi)||p(z|x;\theta)) = 0 \iff q(z;\phi) = p(z|x;\theta)$

- Expectation maximization is precisely equivalent to coordinate-ascent on the ELBO, alternating between ϕ and θ steps

Evidence Lower BOund (ELBO)/Variational Bound

- In order to estimate the likelihood of the entire dataset $\mathcal{D}$, we need $\sum_{i=1}^{N} \log p(x_i; \theta)$
- Summing up over datapoints we get:

$$\max_{\theta} \sum_{i=1}^{N} \log p(x_i; \theta) \geq \max_{\theta, \phi_1, \ldots, \phi_N} \underbrace{\sum_{i=1}^{N} \mathcal{L}(x_i; \theta, \phi_i)}_{ELBO}$$

- Note that we use a *different* ϕ_i for every data point

Summary

Learning as Optimization

Variational learning turns learning into an optimization problem, namely:

$$\max_{\theta,\phi_1,\ldots,\phi_N} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi_i)$$

Summary

Optimal q

The optimal $q(z;\phi)$ used in the bound corresponds to the intractable posterior distribution $p(z|x)$

Summary

Approximating the Posterior

The better $q(z;\phi)$ can approximate the posterior, the smaller $KL(q(z;\phi)||p(z|x))$ we can achieve, the closer ELBO will be to $\log p(x;\theta)$

Computing Gradients of Expectations

- How do you estimate $\nabla_\phi \mathcal{L}$, i.e. $\nabla_\phi \mathbb{E}_{q(z;\phi)}[\log p(x,z) - \log q(z;\phi)]$?
- Define

$$g(z,\phi) = \log p(x,z) - \log q(z;\phi)$$

- What is $\nabla_\phi \mathcal{L}$?

$$\begin{aligned}
\nabla_\phi \mathcal{L} &= \nabla_\phi \int q(z;\phi) g(z,\phi) dz \qquad \text{(chain rule of derivatives)} \\
&= \int \nabla_\phi q(z;\phi) g(z,\phi) + q(z;\phi) \nabla_\phi g(z,\phi) dz \\
&= \int q(z;\phi) \nabla_\phi \log q(z;\phi) g(z,\phi) + q(z;\phi) \nabla_\phi g(z,\phi) dz \\
&= \mathbb{E}_{q(z;\phi)}[\nabla_\phi \log q(z;\phi) g(z,\phi) + \nabla_\phi g(z,\phi)]
\end{aligned}$$

Using $\nabla_\phi \log q = \frac{\nabla_\phi q}{q}$

Score Function Estimator

Recall

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(z;\phi)}[\nabla_\phi \log q(z;\phi) g(z,\phi) + \nabla_\phi g(z,\phi)]$$

Simplify:

$$\begin{aligned}
\mathbb{E}_{q(z;\phi)}[\nabla_\phi g(z,\phi)] &= \mathbb{E}_q[\nabla_\phi[\log p(x,z) - \log q(z;\phi)]] \\
&= \mathbb{E}_q[0 - \frac{\nabla_\phi q(z;\phi)}{q(z;\phi)}] = -\int_z \nabla_\phi q(z;\phi) \\
&= -\nabla_\phi \int_z q(z;\phi) = -\nabla_\phi 1 = 0.
\end{aligned}$$

Gives the gradient:

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(z;\phi)}[\nabla_\phi \log q(z;\phi)(\log p(x,z) - \log q(z;\phi))]$$

Sometimes called likelihood ratio or REINFORCE gradients

[Glynn 1990; Williams, 1992; Wingate+ 2013; Ranganath+ 2014; Mnih+ 2014]

Noisy Unbiased Gradients

Gradient: $\mathbb{E}_{q(z;\phi)}[\nabla_\phi \log q(z;\phi)(\log p(x,z) - \log q(z;\phi))]$

Noisy unbiased gradients with Monte Carlo!

$$\frac{1}{S}\sum_{s=1}^{S} \nabla_\phi \log q(z_s;\phi)(\log p(x,z_s) - \log q(z_s;\phi)),$$

$$\text{where } z_s \sim q(z;\phi)$$

Compiling inference with recognition networks

- Previous learning objective:

$$\max_{\theta,\phi_1,\ldots,\phi_N} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi_i)$$

Compiling inference with recognition networks

- Previous learning objective:

$$\max_{\theta,\phi_1,\ldots,\phi_N} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi_i)$$

- **Key idea:** use a single, *conditional*, variational approximation for all datapoints:

$$q_\phi(z|x)$$

- For every x we get a different posterior approximation

Compiling inference with recognition networks

- Previous learning objective:

$$\max_{\theta,\phi_1,\ldots,\phi_N} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi_i)$$

- **Key idea:** use a single, *conditional*, variational approximation for all datapoints:

$$q_\phi(z|x)$$

- For every x we get a different posterior approximation
- New learning objective:

$$\max_{\phi,\theta} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi)$$

ELBO

$$\mathcal{L}(x;\theta,\phi) = \int_z q_\phi(z|x) \log \frac{p_\theta(x,z)}{q_\phi(z|x)}$$

$$= \int_z q_\phi(z|x) \log p_\theta(x|z) - \int_z q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z)}$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z)]}_{\text{Expectation of Joint Distribution}} + \underbrace{\mathrm{H}(q_\phi(z|x))}_{\text{Entropy of } q_\phi(z|x)}$$

Key Points

Parametric $q(z|x;\phi)$

We're going to learn a conditional parametric approximation $q(z|x;\phi)$ to $p(z|x)$, the posterior distribution.

Shared ϕ

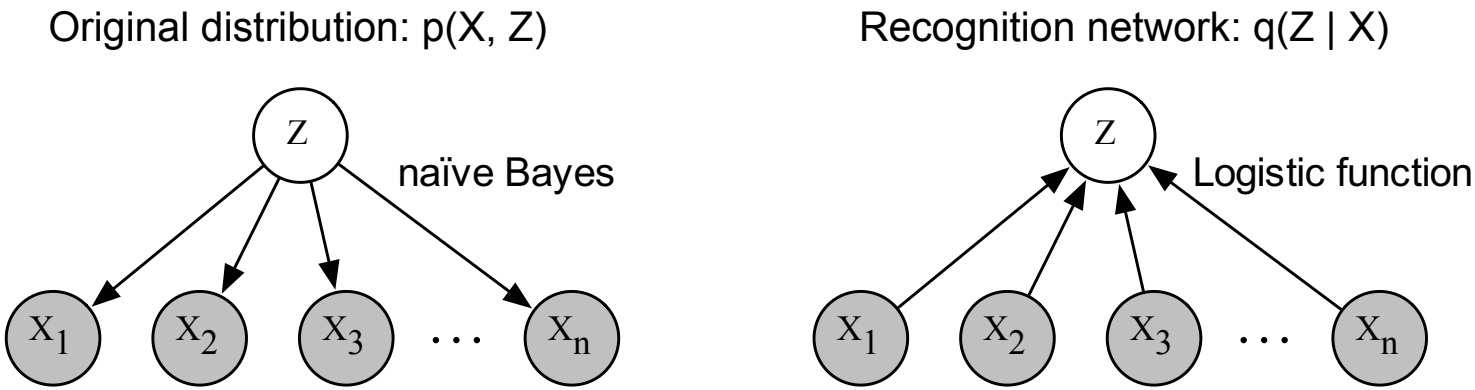
Learning a conditional model, $q(z|x;\phi)$ where ϕ will be shared for all x

Stochastic gradient ascent

We're going to perform joint optimization of θ, ϕ on $\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x_i;\theta,\phi)$

Illustrating the Power of Recognition Networks

- Consider the special case of learning a naïve Bayes model where Z is always latent
- By using a recognition / inference network, we reduce the number of variational parameters from #datapoints to #features (n)



- Because the posterior in naïve Bayes (with binary variables) is a log-linear function, this is without loss of generality!

Variational Auto-encoders

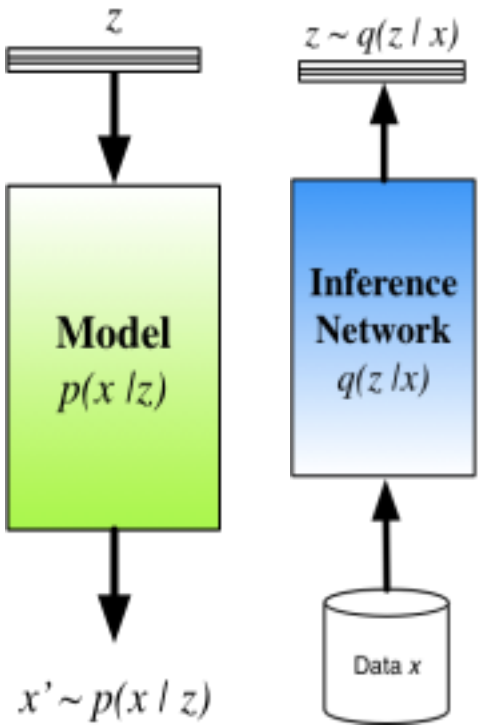
Simplest instantiation of a VAE

Deep Latent Gaussian Model p(x,z)

$$\begin{aligned} \text{prior sample} \quad & z \sim \mathcal{N}(0, \mathbb{I}) \\ \text{data sufficient statistics} \quad & \eta = f_\theta(z) \\ \text{data conditional likelihood} \quad & x \sim \mathcal{N}(\eta) \end{aligned}$$

Gaussian Recognition Model q(z)

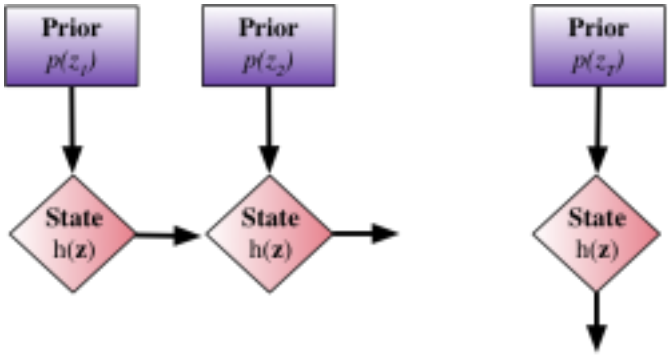
$$\begin{aligned} \text{data sample} \quad & x \sim \mathcal{D} \\ \text{latent sufficient statistics} \quad & \eta = f_\phi(x) \\ \text{posterior sample} \quad & z \sim \mathcal{N}(\eta) \end{aligned}$$



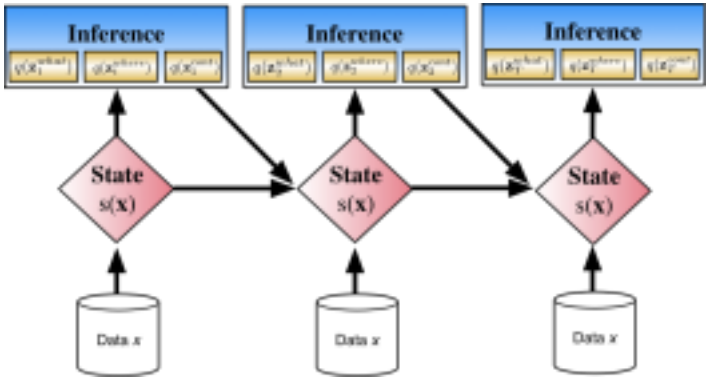
Kingma and Welling, 2014, Rezende et al., 2014

Richer VAES

DRAW: Recurrent/Dependent Priors



Recurrent/Dependent Inference Networks



Stochastic Gradient Estimators

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

Score-function estimator:
Differentiate the density $q(z|x)$

Pathwise gradient estimator:
Differentiate the function $f(z)$

Typical problem areas:

- Generative models and inference
- Reinforcement learning and control
- Operations research and inventory control
- Monte Carlo simulation
- Finance and asset pricing
- Sensitivity estimation

Fu, 2006

Score Function Estimators

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

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

Gradient reweighted by the value of the function (***derivation in earlier slides***)

Other names:

- Likelihood-ratio trick
- Radon-Nikodym derivative
- REINFORCE and policy gradients
- Automated inference
- Black-box inference

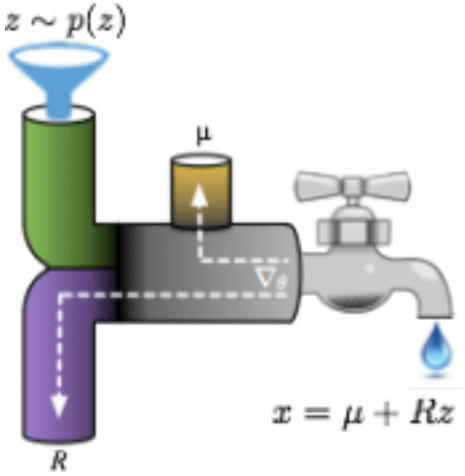
When to use:

- Function is not differentiable.
- Distribution *q* is easy to sample from.
- Density *q* is known and differentiable.

Reparameterisation

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

Find an invertible function *g(.)* that expresses z as a transformation of a base distribution .

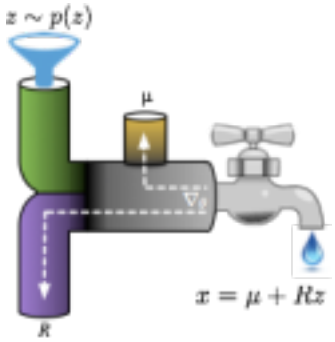


$$\mathbf{z} = g_\phi(\boldsymbol{\epsilon}) \qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(x, \epsilon))]$$

Kingma and Welling, 2014, Rezende et al., 2014

Pathwise Derivative Estimator



$$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$$

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

$$= \mathbb{E}_{p(\epsilon)}[\nabla_\phi f_\theta(g(\epsilon, \phi))]$$

Other names:

- Reparameterisation trick
- Stochastic backpropagation
- Perturbation analysis
- Affine-independent inference
- Doubly stochastic estimation
- Hierarchical non-centred parameterisations.

When to use

Function *f* is differentiable

Density *q* can be described using a simpler base distribution: inverse CDF, location-scale transform, or other co-ordinate transform.

Easy to sample from base distribution.

Gaussian Stochastic Gradients

$$\nabla_\phi \mathbb{E}_{\mathcal{N}(\mu, CC^\top)}[f_\theta(\mathbf{z})]$$

First-order Gradient

$$p(\epsilon) = \mathcal{N}(0,1) \quad g(\epsilon, \phi) = \mu_\phi(x) + C_\phi(x)\epsilon$$

$$\mathbb{E}_{p(\epsilon)}[J^\top(\nabla_\phi \mu_\phi + \nabla_\phi C_\phi^\top \epsilon)]$$

We can develop low-variance estimators by exploiting knowledge of the distributions involved when we know them

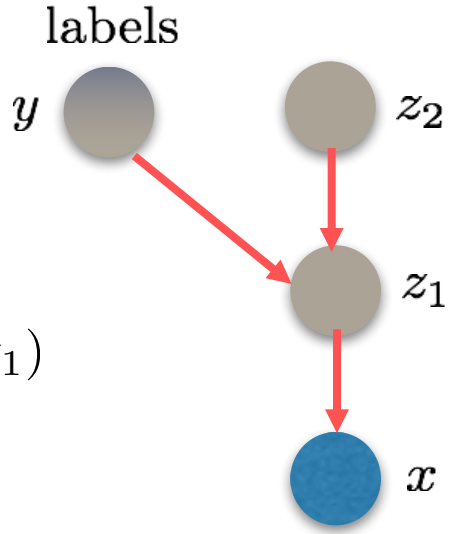
Rezende et al., 2014

Beyond Fully Unsupervised

Extension to Semi-Supervised Learning

- "Semi-supervised Learning with Deep Generative Networks", Kingma et al,'14.

- Labels are treated as either observed or hidden.



$$p(x, y, z_1, z_2) = p(y)p(z_2)p(z_1|z_2, y)p(x|z_1)$$

Extension to Semi-Supervised Learning

- "Semi-supervised Learning with Deep Generative Networks", Kingma et al,'14.

- For datapoint with labels:

$$\log p_\theta(x, y) \geq \mathbb{E}_{q_\beta(z|x,y)} \left(\log p_\theta(x|y, z) + \log p_\theta(y) + \log p(z) - \log q_\beta(z|x, y)\right)$$

- For datapoint with no labels:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\beta(y,z|x)} \left(\log p_\theta(x|y, z) + \log p_\theta(y) + \log p(z) - \log q_\beta(z, y|x)\right)$$

Extension to Semi-Supervised Learning

- *"Semi-supervised Learning with Deep Generative Networks", Kingma et al,'14.*

- Classification results on MNIST:

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

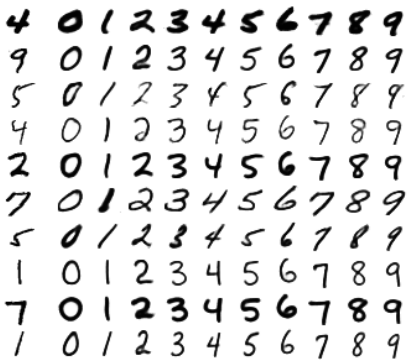
N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 (± 0.95)	11.82 (± 0.25)	11.97 (± 1.71)	**3.33** (± 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 (± 0.049)	4.94 (± 0.13)	**2.59** (± 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 (± 0.12)	4.24 (± 0.07)	3.60 (± 0.56)	**2.40** (± 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 (± 0.04)	3.92 (± 0.63)	**2.18** (± 0.04)

- Now there are stronger models on that task.
  - Ladder-Networks
  - GANs.

Extension to Semi-Supervised Learning

- *"Semi-supervised Learning with Deep Generative Networks", Kingma et al,'14.*
- Disentangling label and "style":

(a) Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable **z**

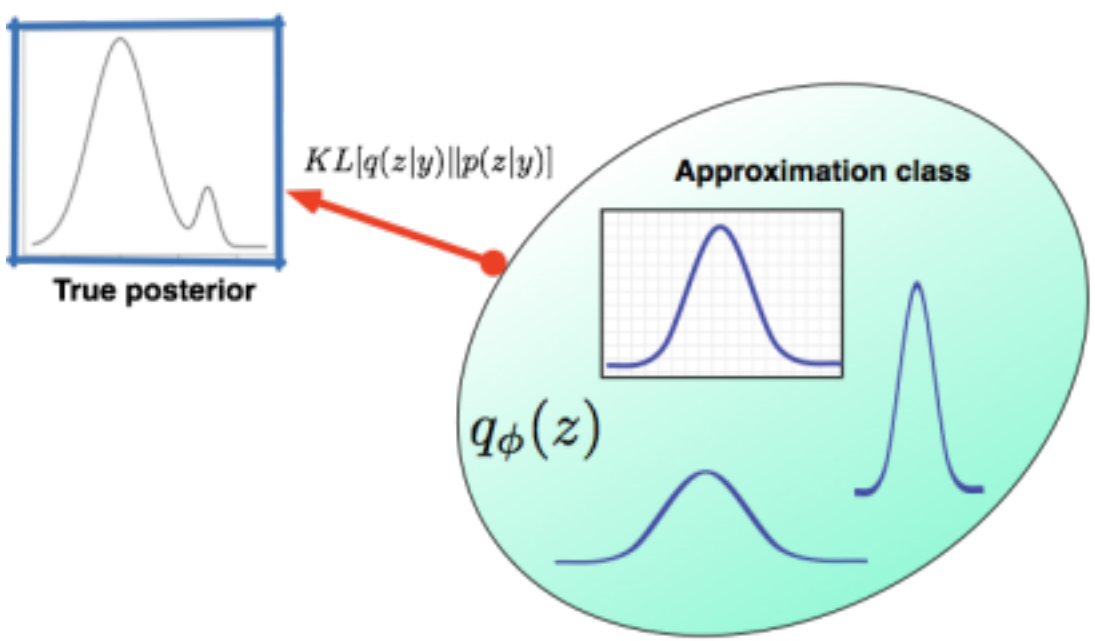


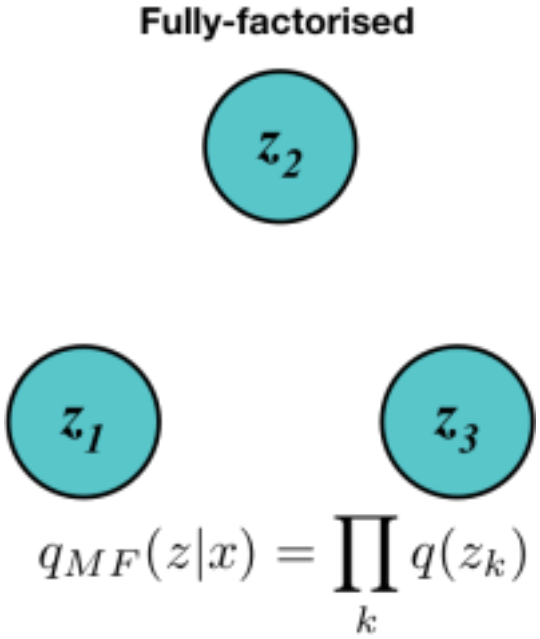
(b) MNIST analogies

(c) SVHN analogies

Beyond the Mean Field

Mean Field Approximations



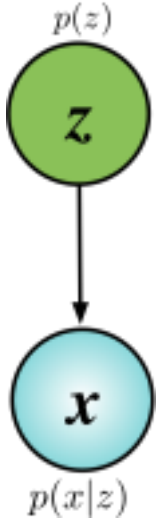


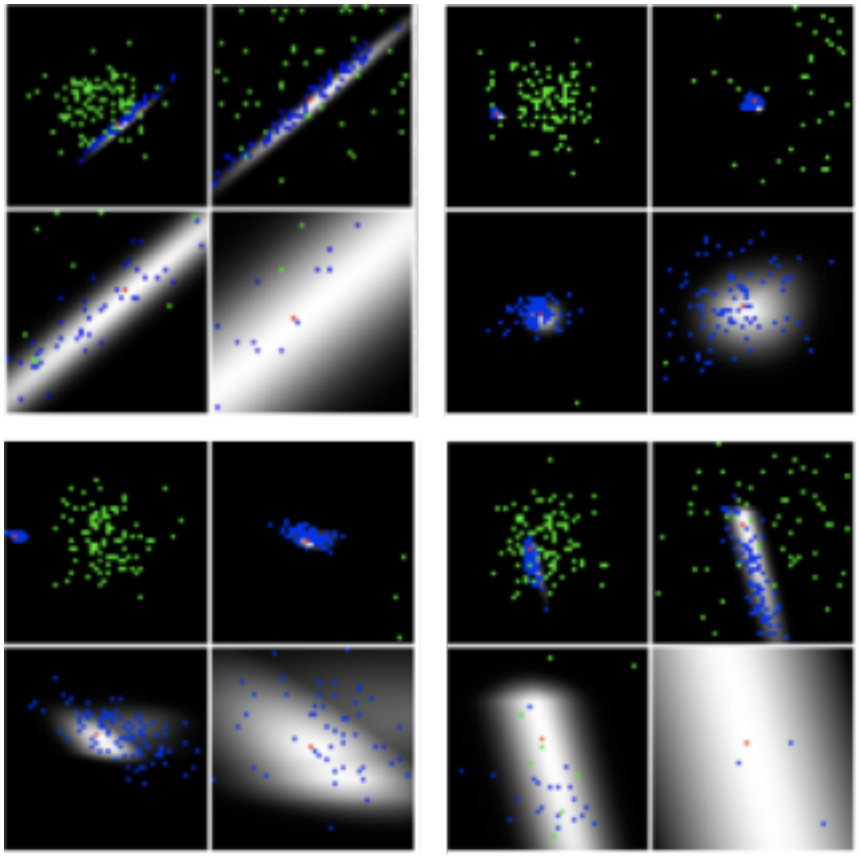
Key part of variational inference is choice of approximate posterior distribution q.

$$\mathcal{F}(q, \boldsymbol{\theta}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}|)\|p(\mathbf{z})]$$

Real-world Posterior Distributions

Deep Latent Gaussian Model



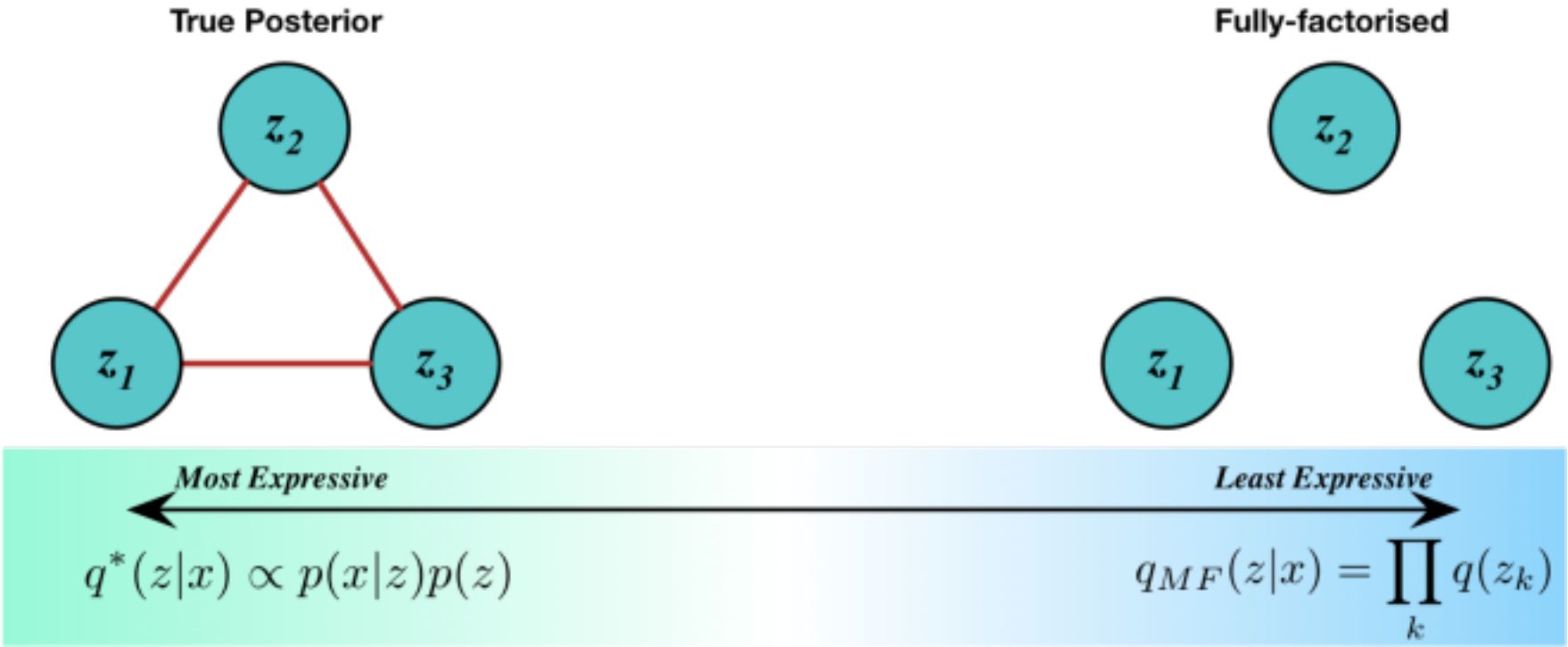


Complex dependencies · Non-Gaussian distributions · Multiple modes

Richer Families of Posteriors

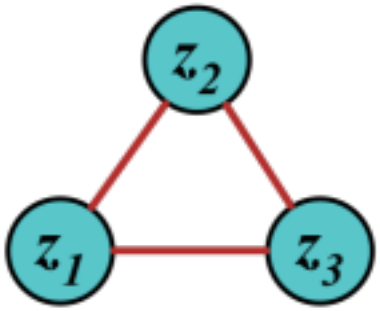
Two high-level goals:

- Build richer approximate posterior distributions.
- Maintain computational efficiency and scalability.



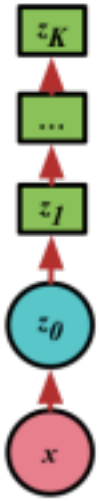
Same as the problem of specifying a model of the data itself

True Posterior

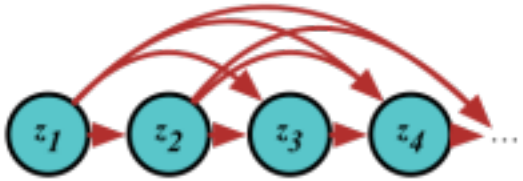


Families of Posterior Approximations

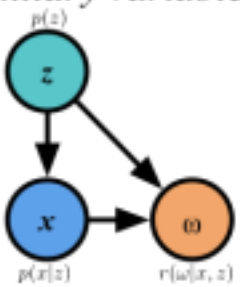
Normalising flows



Structured mean-field

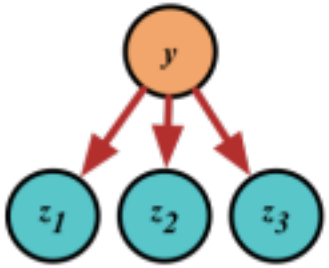


Auxiliary variables

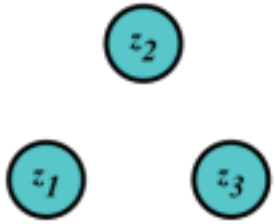


Covariance models

Mixtures



Fully-factorised



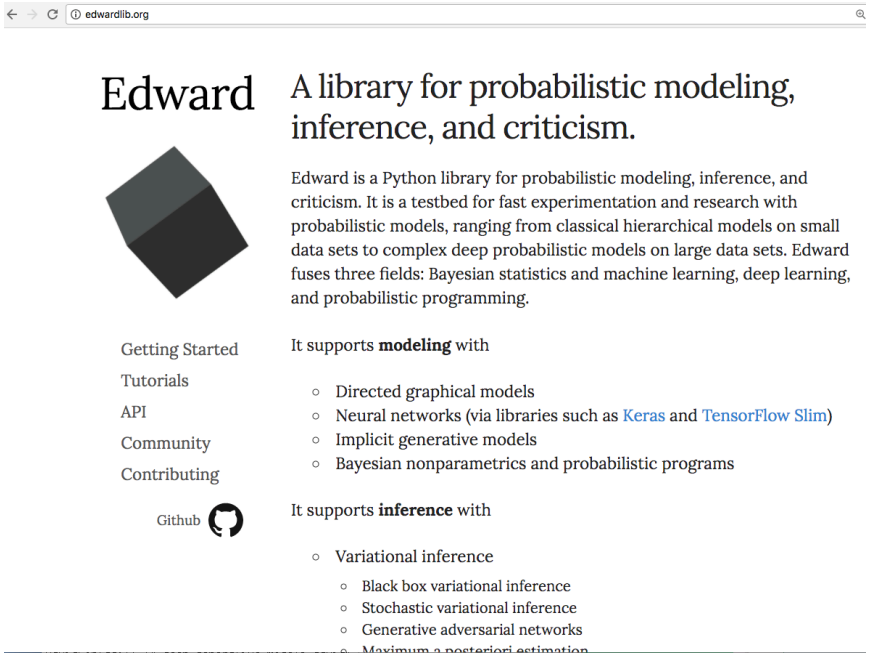
Most Expressive ⟵ ⟶ ***Least Expressive***

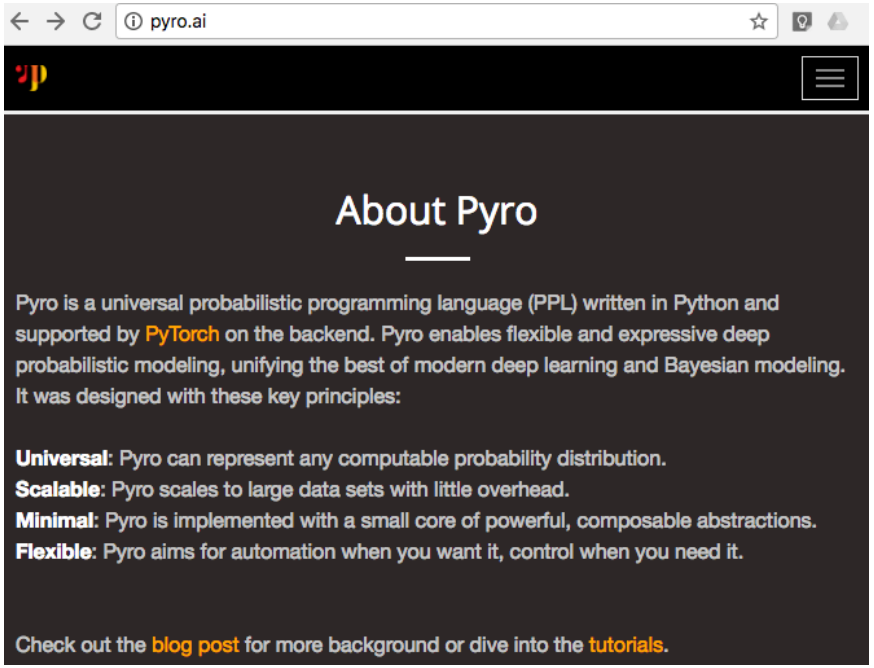
$$q^*(z|x) \propto p(x|z)p(z)$$

$$q_{MF}(z|x) = \prod_k q(z_k)$$

Software for Learning DGMs

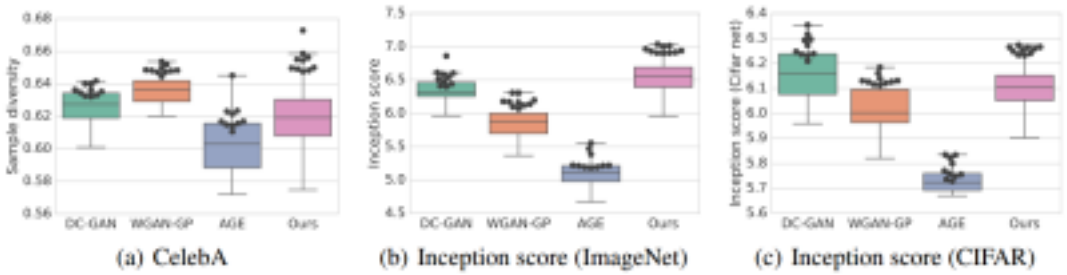
- Learning can be easily implemented as extensions of Tensorflow or Pytorch
- See, in particular, edwardlib.org and pyro.ai for two platforms to build from





Challenges

- Scalability to large images, videos, multiple data modalities.
- Evaluation of generative models.
- Robust conditional models.
- Discrete latent variables.
- Support-coverage in models, mode-collapse.
- Calibration.
- Parameter uncertainty.
- Principles of likelihood-free inference.



(a) CelebA (b) Inception score (ImageNet) (c) Inception score (CIFAR)

References: Fully-observed Models

- Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).
- Larochelle, Hugo, and Iain Murray. "The Neural Autoregressive Distribution Estimator." In AISTATS, vol. 1, p. 2. 2011.
- Uria, Benigno, Iain Murray, and Hugo Larochelle. "A Deep and Tractable Density Estimator." In ICML, pp. 467-475. 2014.

References: Stochastic Gradients & Amortised Inference

- Dayan, Peter, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. "The helmholtz machine." Neural computation 7, no. 5 (1995): 889-904.
- Gershman, Samuel J., and Noah D. Goodman. "Amortized inference in probabilistic reasoning." In Proceedings of the 36th Annual Conference of the Cognitive Science Society. 2014.
- Danilo Jimenez Rezende, Shakir Mohamed, Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." ICML (2014).
- Durk Kingma and Max Welling. "Auto-encoding Variational Bayes." ICLR (2014).
- Pierre L'Ecuyer, Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators, Management Science, 1995
- Rajesh Ranganath, Sean Gerrish, and David M. Blei. "Black Box Variational Inference." In AISTATS, pp. 814-822. 2014.
- Andriy Mnih, and Karol Gregor. "Neural variational inference and learning in belief networks." arXiv preprint arXiv:1402.0030 (2014).

References: Structured Mean Field

- Jaakkola, T. S., and Jordan, M. I. (1998). Improving the mean field approximation via the use of mixture distributions. In Learning in graphical models (pp. 163-173). Springer Netherlands.
- Saul, L.K. and Jordan, M.I., 1996. Exploiting tractable substructures in intractable networks. Advances in neural information processing systems, pp.486-492.
- Gregor, Karol, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. "DRAW: A recurrent neural network for image generation." ICML (2015).
- Gershman, S., Hoffman, M. and Blei, D., 2012. Nonparametric variational inference. arXiv preprint arXiv:1206.4665.
- Felix V. Agakov, and David Barber. "An auxiliary variational method." NIPS (2004). □ Rajesh Ranganath, Dustin Tran, and David M. Blei. "Hierarchical Variational Models." ICML (2016). □ Lars Maaløe et al. "Auxiliary Deep Generative Models." ICML (2016).
- Tim Salimans, Durk Kingma, Max Welling. "Markov chain Monte Carlo and variational inference: Bridging the gap. In International Conference on Machine Learning." ICML (2015).
- Maaløe L, Sønderby CK, Sønderby SK, Winther O. Auxiliary deep generative models. arXiv preprint arXiv:1602.05473. 2016 Feb 17.

References: Normalising Flows

- Tabak, E. G., and Cristina V. Turner. "A family of nonparametric density estimation algorithms." Communications on Pure and Applied Mathematics 66, no. 2 (2013): 145-164.
- Rezende, Danilo Jimenez, and Shakir Mohamed. "Variational inference with normalizing flows." ICML (2015).
- Kingma, D.P., Salimans, T. and Welling, M., 2016. Improving variational inference with inverse autoregressive flow. arXiv preprint arXiv:1606.04934.
- Dinh, L., Sohl-Dickstein, J. and Bengio, S., 2016. Density estimation using Real NVP. arXiv preprint arXiv:1605.08803.