# Bayesian Deep Learning

Kris Sankaran

Nepal Winter School in AI

December 26, 2018

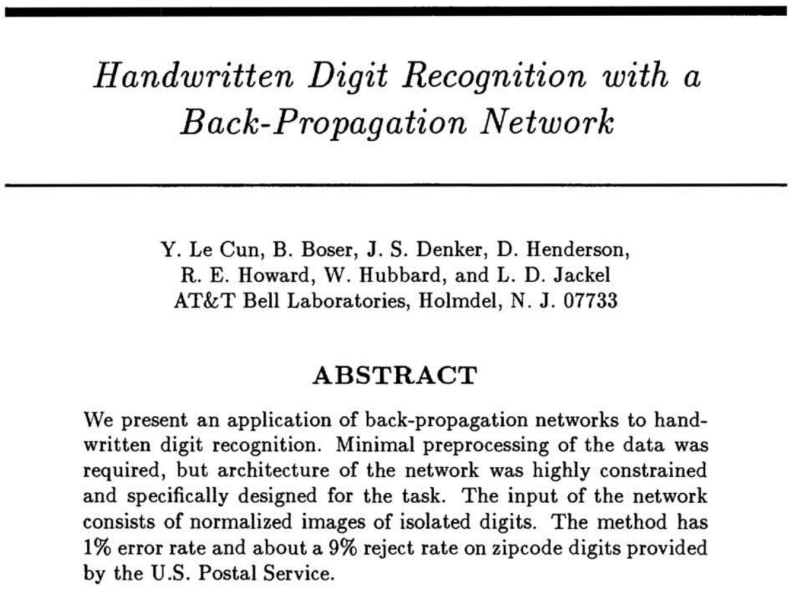
# Outline

# Learning Objectives

- Trace historical development of Bayesian ML, from Variational Inference to Bayesian Deep Learning
- Add some more models and algorithm to personal catalog of examples
- Understand underlying math and code for example algorithms

# Papers from 1990

- Gelfand and Smith: Renaissance in Bayesian Inference
- LeCun et. al.: Early proof of representation learning through neural networks
- Common: Use computation to automate tediour, expert-labor intensive processes
- Tension: The "two cultures"... prediction or inference?

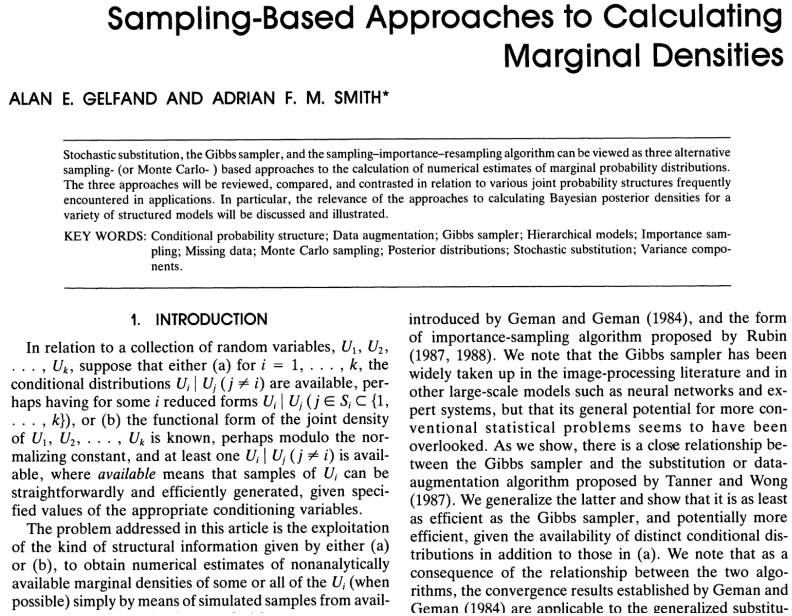


*Handwritten Digit Recognition with a Back-Propagation Network*

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson,
R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

**ABSTRACT**

We present an application of back-propagation networks to handwritten digit recognition. Minimal preprocessing of the data was required, but architecture of the network was highly constrained and specifically designed for the task. The input of the network consists of normalized images of isolated digits. The method has 1% error rate and about a 9% reject rate on zipcode digits provided by the U.S. Postal Service.



Sampling-Based Approaches to Calculating Marginal Densities

ALAN E. GELFAND AND ADRIAN F. M. SMITH*

Stochastic substitution, the Gibbs sampler, and the sampling–importance–resampling algorithm can be viewed as three alternative sampling- (or Monte Carlo- ) based approaches to the calculation of numerical estimates of marginal probability distributions. The three approaches will be reviewed, compared, and contrasted in relation to various joint probability structures frequently encountered in applications. In particular, the relevance of the approaches to calculating Bayesian posterior densities for a variety of structured models will be discussed and illustrated.

KEY WORDS: Conditional probability structure; Data augmentation; Gibbs sampler; Hierarchical models; Importance sampling; Missing data; Monte Carlo sampling; Posterior distributions; Stochastic substitution; Variance components.

1. INTRODUCTION

In relation to a collection of random variables, $U_1, U_2, \ldots, U_k$, suppose that either (a) for $i = 1, \ldots, k$, the conditional distributions $U_i \mid U_j\ (j \neq i)$ are available, perhaps having for some $i$ reduced forms $U_i \mid U_j\ (j \in S_i \subset \{1, \ldots, k\})$, or (b) the functional form of the joint density of $U_1, U_2, \ldots, U_k$ is known, perhaps modulo the normalizing constant, and at least one $U_i \mid U_j\ (j \neq i)$ is available, where *available* means that samples of $U_i$ can be straightforwardly and efficiently generated, given specified values of the appropriate conditioning variables.

The problem addressed in this article is the exploitation of the kind of structural information given by either (a) or (b), to obtain numerical estimates of nonanalytically available marginal densities of some or all of the $U_i$ (when possible) simply by means of simulated samples from avail-

introduced by Geman and Geman (1984), and the form of importance-sampling algorithm proposed by Rubin (1987, 1988). We note that the Gibbs sampler has been widely taken up in the image-processing literature and in other large-scale models such as neural networks and expert systems, but that its general potential for more conventional statistical problems seems to have been overlooked. As we show, there is a close relationship between the Gibbs sampler and the substitution or data-augmentation algorithm proposed by Tanner and Wong (1987). We generalize the latter and show that it is as least as efficient as the Gibbs sampler, and potentially more efficient, given the availability of distinct conditional distributions in addition to those in (a). We note that as a consequence of the relationship between the two algorithms, the convergence results established by Geman and Geman (1984) are applicable to the generalized substitu-

# Reconciliation

- We need both inference and prediction for systems that *sense* and *act* in the real world
  - Need systems running in real time, interfacing with the world
  - Probabilistic descriptions give us much more to base our decisions off of
  - It would be nice if everything were automatic, or at least modular
- Bayesian deep learning: modular feature learning with uncertainty

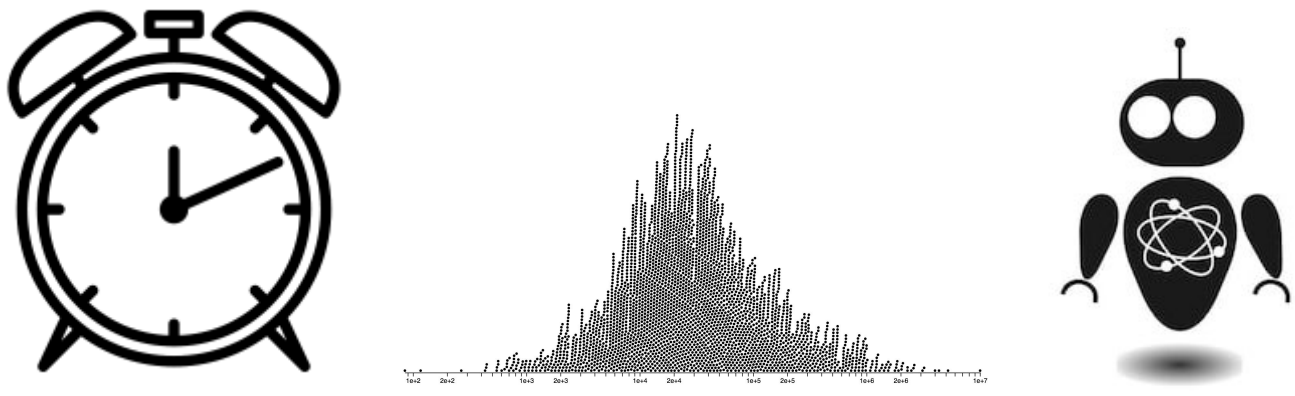


Figure: We would like our modeling to be fast, probabilistic, and automatic.

# Machine Learning checklist

Questions to ask every algorithm you meet.

- What is the model class?
- What is the evaluation criterion?
- What is the search strategy?

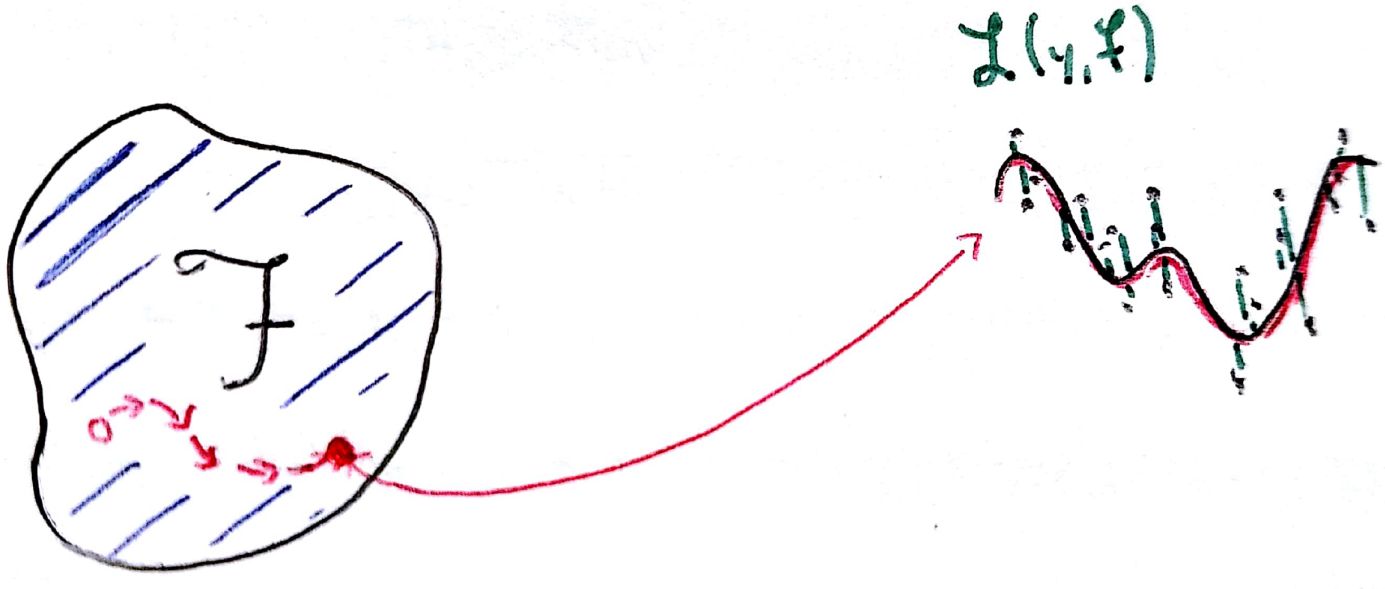



Figure: Each point in $\mathcal{F}$ represents a different function, which we evaluate using $\mathcal{L}$. We have to search through $\mathcal{F}$, represented by the red dashed line.

# In Bayesian Deep Learning

Questions to ask every algorithm you meet.

- What is the model class?
  - Gaussian process?
  - Mixture of multinomials?
  - Gaussian with neural network $\mu_\theta$?
- What is the evaluation criterion?
  - Evidence
  - Evidence Lower Bound
  - Hold-out Log Likelihood
- What is the search strategy?
  - Variational Inference
  - MCMC
  - Expectation Propagation

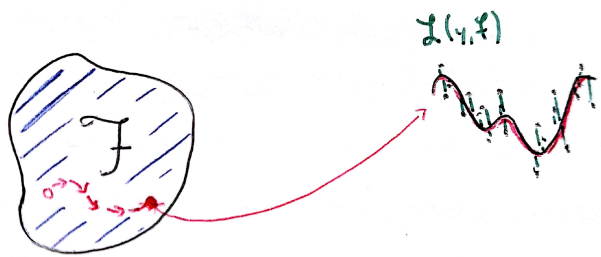



Figure: Each point in $\mathcal{F}$ represents a different function, which we evaluate using $\mathcal{L}$. We have to search through $\mathcal{F}$, represented by the red dashed line.

# Classical Bayes

- Bayes' Rule,

$$p(\theta|x) = \frac{p(x|\theta)\, p(\theta)}{p(x)}$$

- Adapt your belief about $\theta$ after seeing $x$
- Specify likelihood $p(x|\theta)$ and prior $p(\theta)$

# Example: Beta-Binomial

- Task: Identify potential bias in a coin
- Model specification,
  - Prior: $p \sim \text{Beta}\,(a_0, b_0)$
  - Likelihood: $x|p \sim \text{Bin}\,(n, p)$
- Posterior is still a beta

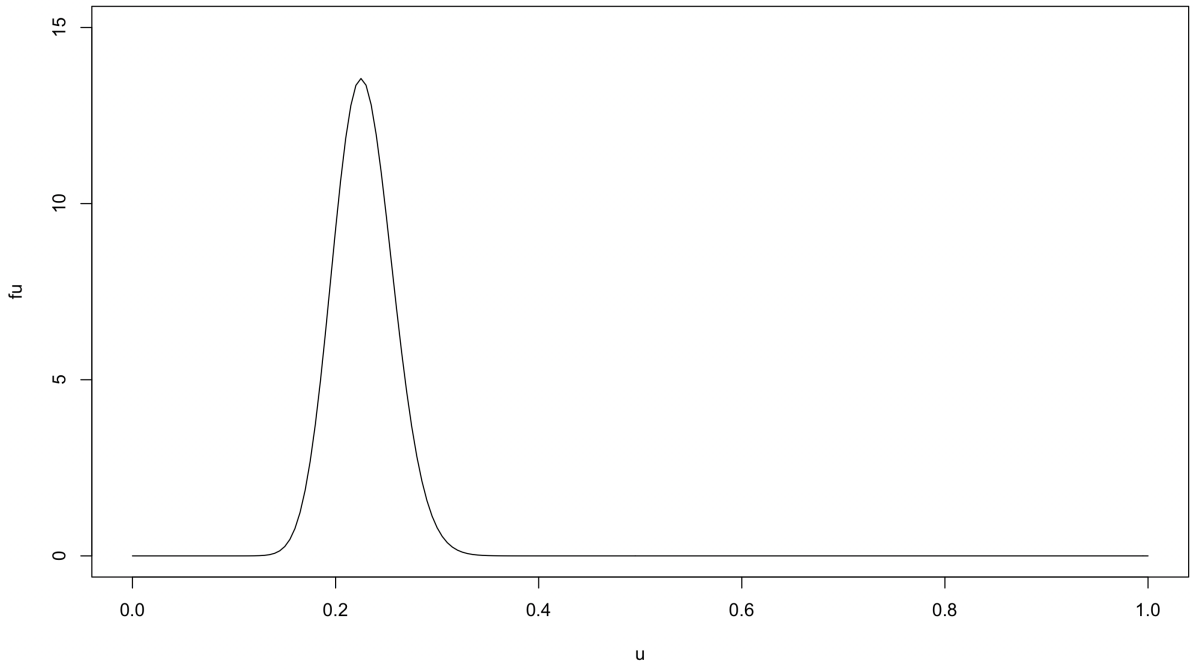



Figure: After seeing more and more coin flips, the posterior becomes more sure that the underlying probability is around 0.2.

# Beta-Binomial in Code

- We can avoid doing any math by using the `pyro` package
- This is good practice for when it's impossible to do that math
- Hinges on the notion of a guide function

```
def model(data):
    # define the hyperparameters that control the beta prior
    alpha0 = torch.tensor(10.0)
    beta0 = torch.tensor(10.0)

    # sample f from the beta prior, then draw from the likelihood
    p = pyro.sample("heads_prob", dist.Beta(alpha0, beta0))
    with pyro.iarange("data", len(data)):
        pyro.sample(
            "obs",
            dist.Bernoulli(p).expand_by(data.shape).independent(1),
            obs=data
        )
```

Figure: Example implementation of beta-binomial model in a probabilistic programming language.

# Beta-Binomial in Code

- We can avoid doing any math by using the `pyro` package
- This is good practice for when it's impossible to do that math
- Hinges on the notion of a guide function

```
def guide(data):
    # register the two variational parameters with Pyro.
    alpha_q = pyro.param(
        "alpha_q",
        torch.tensor(15.0),
        constraint=constraints.positive
    )
    beta_q = pyro.param(
        "beta_q",
        torch.tensor(15.0),
        constraint=constraints.positive
    )
    # sample heads_prob from the distribution Beta(alpha_q, beta_q)
    pyro.sample("heads_prob", dist.Beta(alpha_q, beta_q))
```

Figure: Example implementation of beta-binomial inference in a probabilistic programming language.

# Outline

# Mixture Models

- Sometimes, critical parts of data generating mechanisms are unobserved
- Example: Need posterior inference over both mixture parameters $\mu_k, \Sigma_k$ as well as assignments $z_i$

$$\mu_k \sim \mathcal{N}(0, \Sigma_0)$$
$$z_i \sim \text{Cat}(z_i|\pi)$$
$$x_i|z_i = k \sim \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

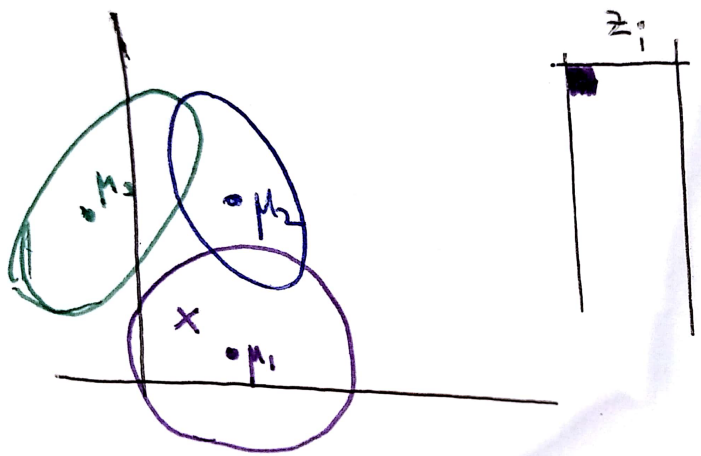



Figure: We've drawn three gaussian densities from the overall prior, and mark an `X` at the first sample, along with its (in reality unobserved) class membership $z_i$.

# Mixture Models

- Sometimes, critical parts of data generating mechanisms are unobserved
- Example: Need posterior inference over both mixture parameters $\mu_k, \Sigma_k$ as well as assignments $z_i$

$$\mu_k \sim \mathcal{N}(0, \Sigma_0)$$
$$z_i \sim \text{Cat}(z_i|\pi)$$
$$x_i|z_i = k \sim \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

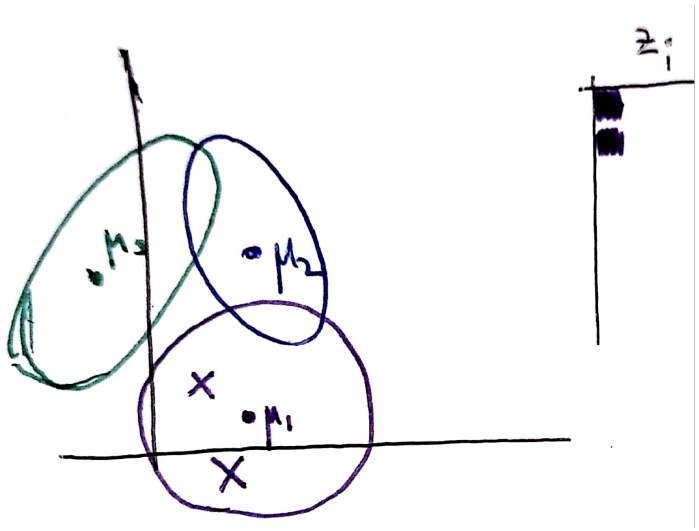



Figure: Draw another point from the same cluster.

# Mixture Models

- Sometimes, critical parts of data generating mechanisms are unobserved
- Example: Need posterior inference over both mixture parameters $\mu_k, \Sigma_k$ as well as assignments $z_i$

$$\mu_k \sim \mathcal{N}(0, \Sigma_0)$$
$$z_i \sim \text{Cat}(z_i|\pi)$$
$$x_i|z_i = k \sim \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

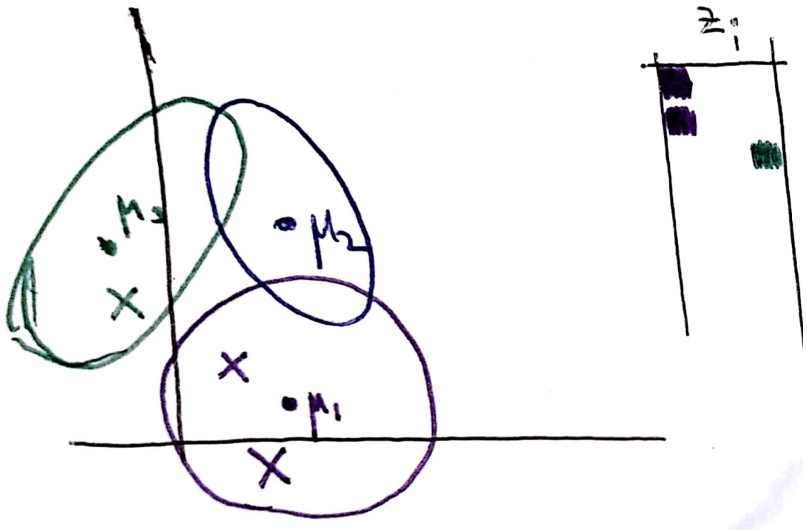



Figure: First point from a new cluster.

# Mixture Models

- Sometimes, critical parts of data generating mechanisms are unobserved
- Example: Need posterior inference over both mixture parameters $\mu_k, \Sigma_k$ as well as assignments $z_i$

$$\mu_k \sim \mathcal{N}(0, \Sigma_0)$$
$$z_i \sim \text{Cat}(z_i|\pi)$$
$$x_i|z_i = k \sim \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

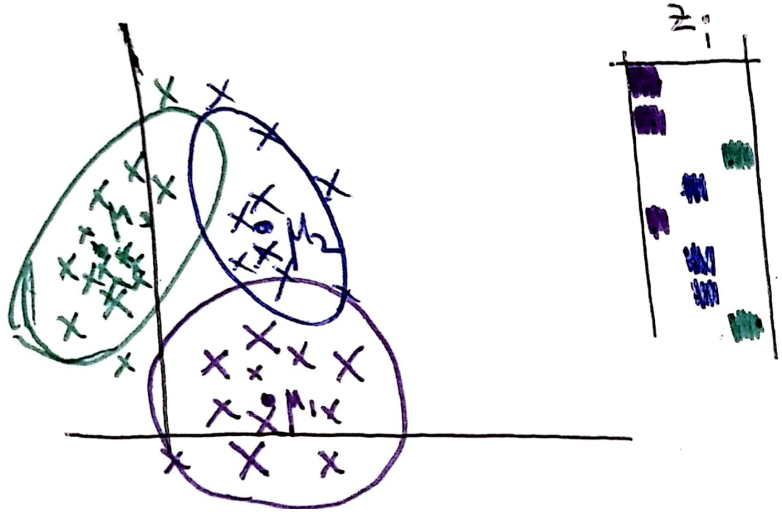


Figure: Continue in this way until you have all $n$ points.

# Other Examples

- Many widely-used models depend on being able to work with latent variables,
  - Exponential Family PCA
  - Hidden Markov Models
  - Latent Dirichlet Allocation
- $z_i$'s can be discrete or continuous
- Latent variables $z_i$ transform simple distributions (Gaussians) into complex ones (mixture of Gaussians)
- In general, mechanism is

$$\theta \sim p_\eta(\theta) \qquad \text{(prior on parameters)}$$
$$z_i \sim p_\varphi(z_i) \qquad \text{(prior on latents)}$$
$$x_i|z_i, \theta \sim p_\theta(x_i|z_i) \qquad \text{(likelihood)}$$

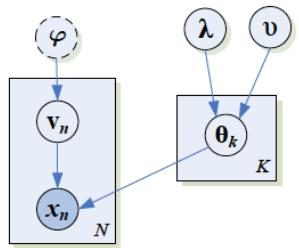



Figure: General form of a latent variable model, as given in [3].

# Mixture of Gaussians: Complete Data Likelihood

If we know the mixture assignments $z_i$ for each sample, we could write out the likelihood very explicitly.

$$
\begin{aligned}
& \log p\left(x, z\right) \\
= & \log\left[\prod_{i=1}^{n}\prod_{k=1}^{K}\left(\pi_k \mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)^{\mathbb{I}(z_i=k)}\right)\right] \\
= & \sum_{i=1}^{n}\sum_{k=1}^{K}\mathbb{I}\left(z_i = k\right)\left[\log \pi_k + \log \mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)\right] \\
\stackrel{c}{=} & \sum_{i,k}\mathbb{I}\left(z_i = k\right)\left[\log \pi_k - \frac{1}{2}\log\left|\Sigma_k\right| - \frac{1}{2}\left(x_i - \mu_k\right)^T \Sigma_k^{-1}\left(x_i - \mu_k\right)\right]
\end{aligned}
$$

# Marginalizing $z$

- How can we fit $\theta$ if we don't know the particular configuration of $z$?
- Way too many configurations to actual do this sum

$$\log p_\theta(x) = \log \sum_z p_\theta(x, z)$$

# Evidence Lower Bound

- It's counterintuitive, but one problem-solving strategy is to deliberately introduce complexity
- Complexity $\rightarrow$ more degrees of freedom

# Variational $q$

- Consider some $q(z|x) \in \mathcal{Q}$, some large family of tractable densities
- Use age-old the "multiply and divide by 1" trick

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_q\left[\log p_\theta(x)\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(x,z)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(x,z)}{q(z|x)} \frac{q(z|x)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_q\left[\log p_\theta(x|z)\right] - D_{KL}\left(q(z|x)\,||p(z)\right) + D_{KL}\left(q(z|x)\,||p_\theta(z|x)\right)
\end{aligned}
$$

# Studying the bound

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_q[\log p_\theta(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p_\theta(z|x)) \\ &\geq \mathbb{E}_q[\log p_\theta(x|z)] - D_{KL}(q(z|x)||p_\theta(z))\end{aligned}$$

- Reconstruction error: How plausible is the observed $x$, averaging over assignments $z$, and considering the current likelihood estimate?
- Approximation complexity: How far is the approximating $q(z|x)$ from the prior?
- Inference quality: How different is the posterior approximation $p(z|x)$ from the actual posterior?
  - $D_{KL} \geq 0$ for any pair of probabilities
  - Hard to compute, so just drop and turn into an inequality

# Discussion

- Consider the collection of points below
- They are a mixture of gaussians with latent $z_i$
- $q\left(z_i|x_i\right)$ gives assignment probabilities to each point
- Which has higher reconstruction error?
- Which has higher approximation complexity?

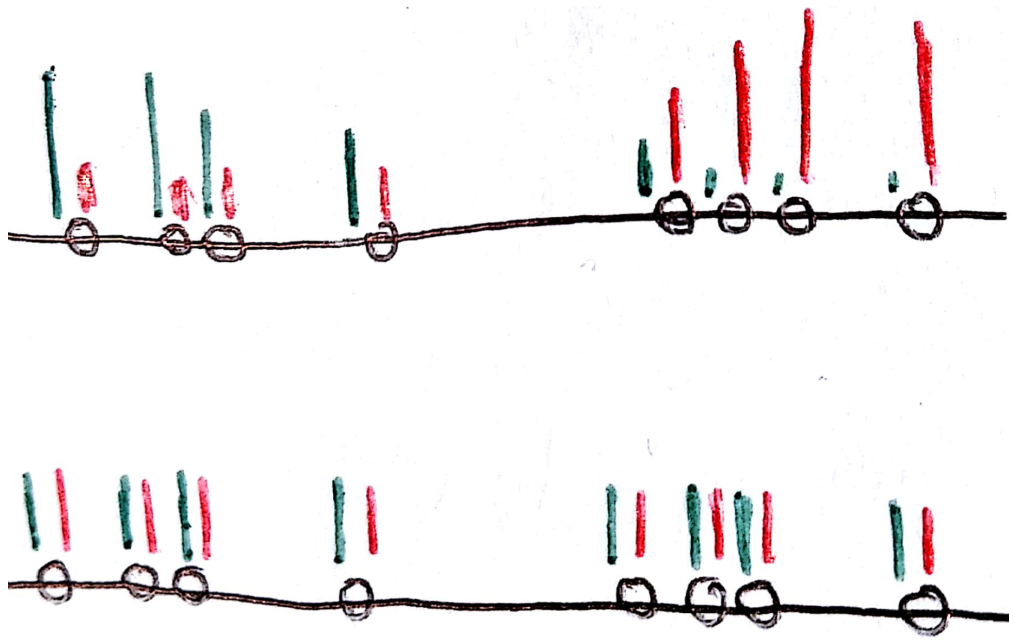


Figure: Each point is an $x_i$. The two vertical bars over each point give a Bernoulli $q\left(z_i|x\right)$ latent assignment probabilities for each point.

# Discussion

- Consider the collection of points below
- $q\left(z_i|x_i\right)$ gives assignment probabilities to each point
- Which has higher reconstruction error?
  - The second distribution has higher error
- Which has higher approximation complexity?
  - The second distribution has higher complexity

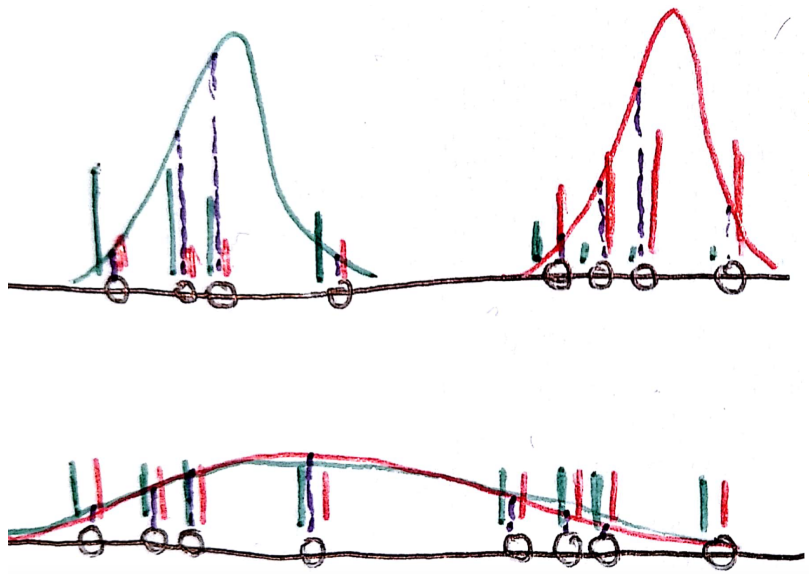


Figure: The $q$'s which separate the components have higher likelihoods (doesn't have to spread all over), and so lower reconstruction errors. However, since Bernoulli's with $p = 0.5$ have the highest entropy, this assignment distribution also has higher approximation complexity.

# Variational Expectation Maximization (EM)

- We now have a strategy for optimizing $\theta$'s
- Alternately update $\theta$ and $q\left(z|x\right)$ to maximize the ELBO
  - E-Step: Find a distribution $q\left(z|x\right)$ over configurations $z$ that is most plausible with the current guess at $\theta$
  - M-Step: Find a set of parameters $\theta$ that is most plausible with the current distribution over configurations $z$
- Hope is that this also increases $p_\theta\left(x\right)$

# Outline

# The Variational Idea

- Transform integration problem into an optimization one
- Some families $\mathcal{Q}$ are easier to optimize over than others

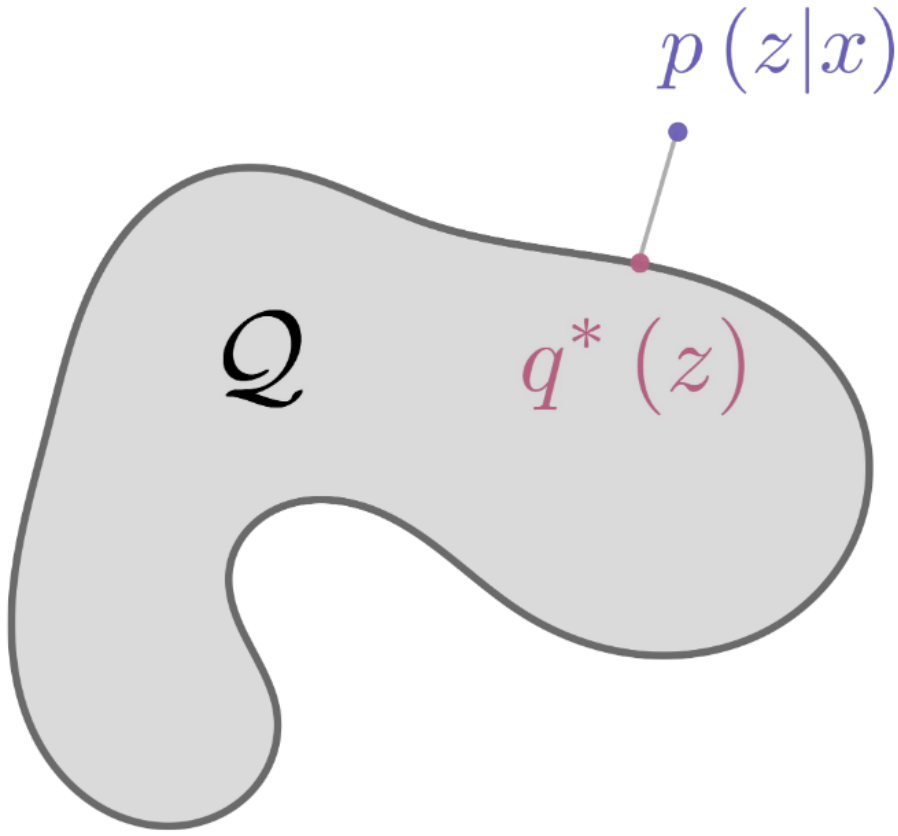



Figure: The hope is to find a density $q^* \in \mathcal{Q}$ that's relatively close to the true posterior.

# The Variational Idea

- Transform integration problem into an optimization one
- Some families $\mathcal{Q}$ are easier to optimize over than others

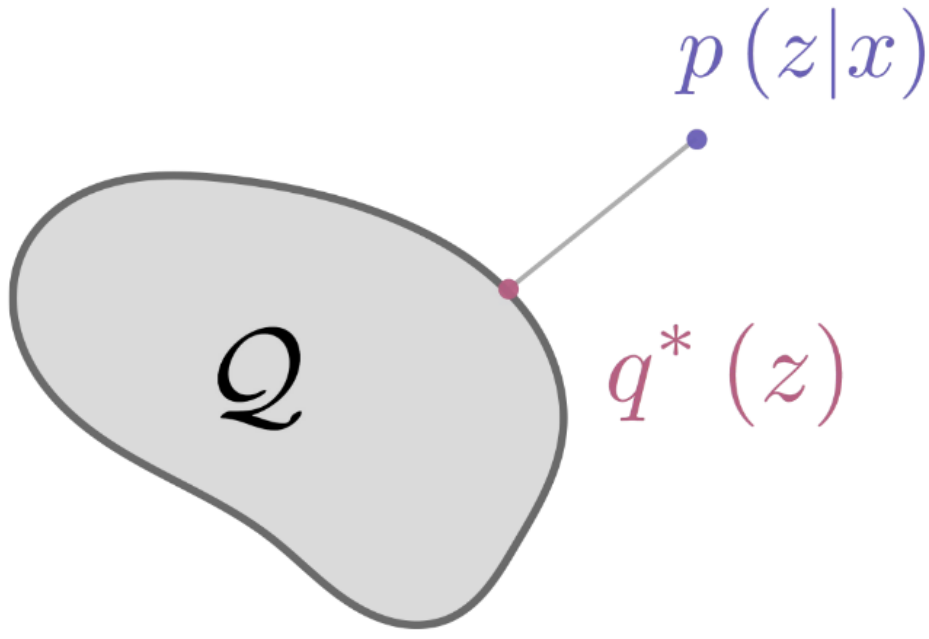



Figure: Some families might be much easier to optimize over than others.

# Doing Variational Inference

To practically implement this idea need to determine

- What densities $\mathcal{Q}$ can we actually work with?
- How are we going to measure the quality of an approximation?

# Doing Variational Inference

To practically implement this idea need to determine

- What densities $\mathcal{Q}$ can we actually work with?
  - Mean-Field, Structured Mean-Field approximations, or even implicit densities
- How are we going to measure the quality of an approximation?
  - ELBO, $f$-divergences, Wasserstein distance, ...

# Doing Variational Inference

To practically implement this idea need to determine

- What densities $\mathcal{Q}$ can we actually work with?
  - **Mean-Field**, Structured Mean-Field approximations, or even implicit densities
- How are we going to measure the quality of an approximation?
  - **ELBO**, $f$-divergences, Wasserstein distance, ...

# Mean-Field Approximation

Consider a family where all the variables factor

$$q(z_1, \ldots, z_n) = \prod_{i=1}^{n} q_i(z_i)$$

(notice that we drop conditioning on $x_i$, this is actually more general than before)

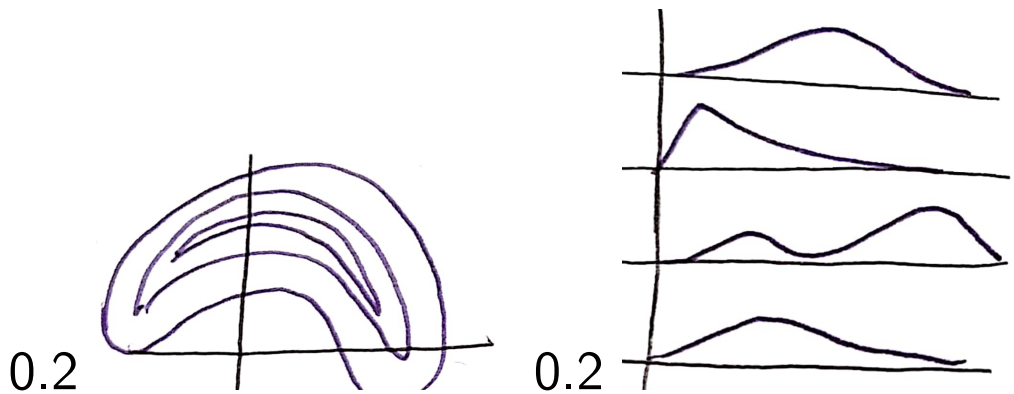



Figure: Instead of trying to model the joint relationships across all the latent variables, we will concern ourselves with one coordinate at a time.

# Mean-Field Approximation

Consider a family where all the variables factor

$$q(z_1, \ldots, z_n) = \prod_{i=1}^{n} q_i(z_i)$$

(notice that we drop conditioning on $x_i$, this is actually more general than before)

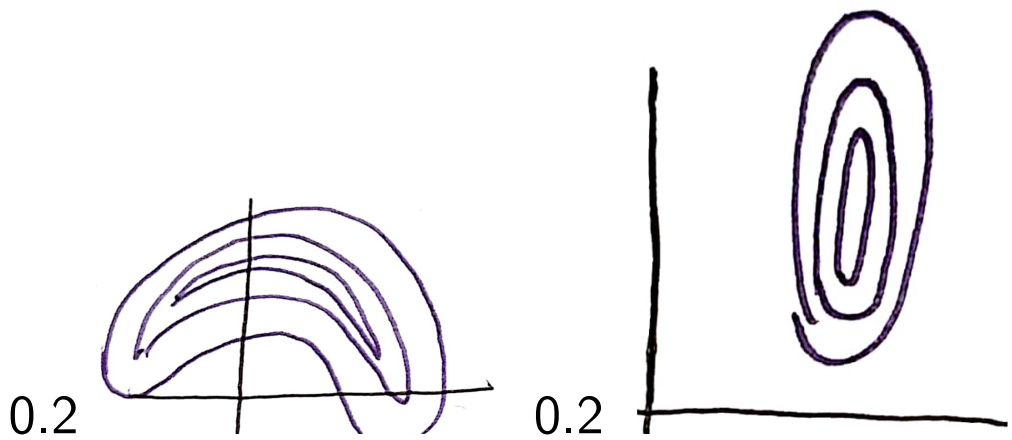



Figure: A consequence of using the product is that we can't model complex correlations between coordinates (everything has to be axis-aligned).

# Optimization

- Fixing all but the $i^{th}$ coordinate's distribution, can find how to optimize ELBO directly

$$q_i(z_i) \propto \exp\left(\mathbb{E}_{q_{-i}}\left[\log p(x, z_i, z_{-i})\right]\right)$$

The proof of this fact is only a few lines (not at all obvious though)

$$\begin{aligned}
\log p(x) &\geq \mathbb{E}_q[\log p(x|z)] - D_{KL}(q(z)||p(z)) \\
&= \mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)] \\
&= \mathbb{E}_{q_i}\left[\mathbb{E}_{q_{-i|i}}[\log p(x,z_i,z_{-i})]\right] - \sum \mathbb{E}_{q_j}[\log q_j(z_j)] \\
&\stackrel{c}{=} -D_{KL}\left(q_i(z_i)\,||\,\exp\left(\mathbb{E}_{q_{-i}}[p(x,z_i,z_{-i})]\right)\right)
\end{aligned}$$

Since KL is always $\geq 0$, can maximize this expression by setting it to zero (set $q_i$ to right hand side distribution)

# Outline

# A Calculated Tradeoff

- It's nice that this approach works for arbitrary mean-field $\prod q_i(z_i)$
- But it's restrictive to have to compute

$$q_i(z_i) \propto \exp\left(\mathbb{E}_{q_{-i}}\left[\log p(x, z_i, z_{-i})\right]\right)$$

in closed form (not to mention tedious)

  - How could we have (say) $p(x|z) = \mathcal{N}\left(x|\mu_\theta(z), \sigma_\theta^2(x)\right)$ for some complicated $\mu_\theta, \sigma_\theta^2$?

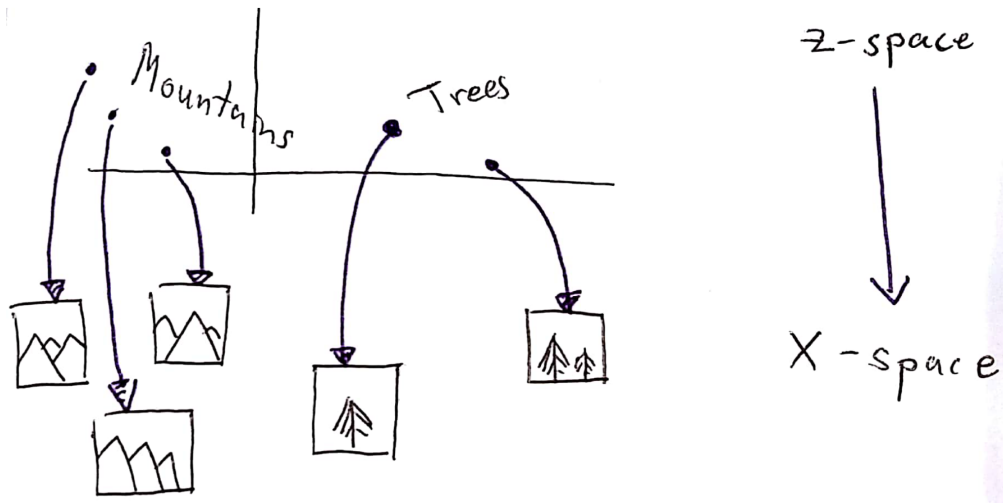



Figure: A flexible generative mechanism will let us map latent $z_i$'s into complex distributions in the $x_i$ space. This process is sometimes called "decoding," because it takes an unobserved $z$ and transforms it into some $x$ living in the observation space.

# Inference Networks

- Idea: Restrict family $\mathcal{Q}$ (hopefully not too much), and make optimization more automatic
- New approximating family: $q_\varphi(z|x) = \mathcal{N}\left(z|\mu_\varphi(x), \sigma^2_\varphi(x) I\right)$
- Let $\mu_\varphi$ and $\sigma^2_\varphi$ be arbitrary nonlinear functions

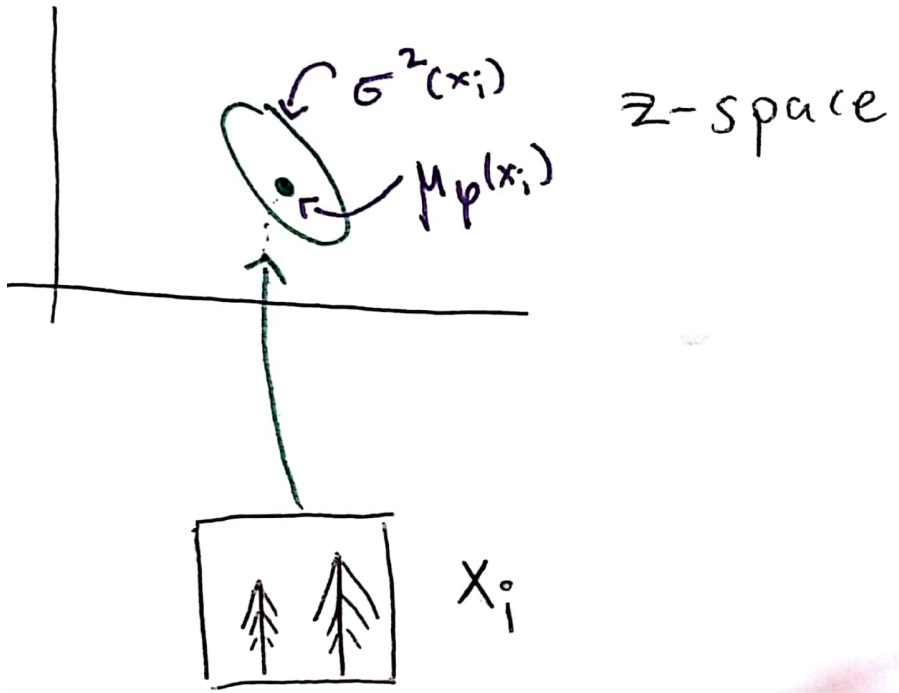



Figure: Each $x_i$ is associated with it's own distribution in the $z_i$ space. The parameters of this distribution are determined by the inference network. This process is sometimes called "encoding," because it encodes an observed $x$ into a latent $z$.

# Optimization

- Instead of optimizing one coordinate at a time, optimize all at once through $\varphi$
- Simultaneously optimize parameters $\theta$
- Essentially variational EM, but where $q$ is a parameterized function of the $x_i$

# Reparametrization Trick

- Taking gradient step along

$$\nabla_{\varphi} \mathbb{E}_q \left[ \log p_\theta \left( x|z \right) \right]$$

  is complicated, because

$$\nabla_{\varphi} \mathbb{E}_q \left[ \log p_\theta \left( x|z \right) \right] = \int \log p_\theta \left( x|z \right) \nabla_{\varphi} q_\varphi \left( z|x \right) dz$$

  is no longer an expectation over $q_\varphi$.
- Can't just sample, and can't do integral analytically.

# Reparametrization Trick

- Sampling

$$z|x \sim q_\varphi\left(z|x\right)$$

  is sometimes the same as

$$\epsilon \sim p_0\left(\epsilon\right)$$
$$z|\epsilon, x \equiv g_\varphi\left(x, \epsilon\right)$$

  for some deterministic $g_\varphi$

- Large family of densities $\rightarrow$ one density with large family of transformations

# Reparametrization Trick

Most common example, if you want to sample

$$z|x \sim \mathcal{N}\left(z|\mu_\varphi\left(x\right), \sigma^2_\varphi\left(x\right)\right)$$

instead use

$$\epsilon \sim \mathcal{N}\left(0, I\right)$$
$$z|x, \epsilon \equiv \mu_\varphi\left(x\right) + \sigma^2_\varphi\left(x\right) \odot \epsilon$$

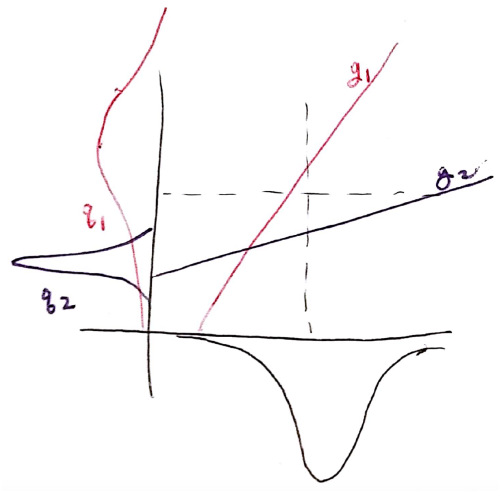


Figure: Instead of working with multiple gaussian densities (purple and red gaussians), we work with one and transform it with different linear transformations.

# Optimization after Reparameterization

- Since $g_\varphi$ is deterministic, gradient can be easily approximated,

$$\begin{aligned}\nabla_\varphi \mathbb{E}_{q_\varphi}\left[\log p_\theta\left(x|z\right)\right] &= \nabla_\varphi \mathbb{E}_{p(\epsilon)}\left[\log p_\theta\left(x|g_\varphi\left(\epsilon\right)\right)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\nabla_\varphi \log p_\theta\left(x|g_\varphi\left(\epsilon\right)\right)\right] \\ &\approx \sum_i \nabla_\varphi \log_\theta p_\theta\left(x|g_\varphi\left(\epsilon_i\right)\right)\end{aligned}$$

  after sampling lots of $\epsilon_i \sim p\left(\epsilon\right)$.
- Derivative of $D_{KL}$ term can usually be written in closed form
- We can optimize the ELBO!

# Outline

# Latent Space & Generation

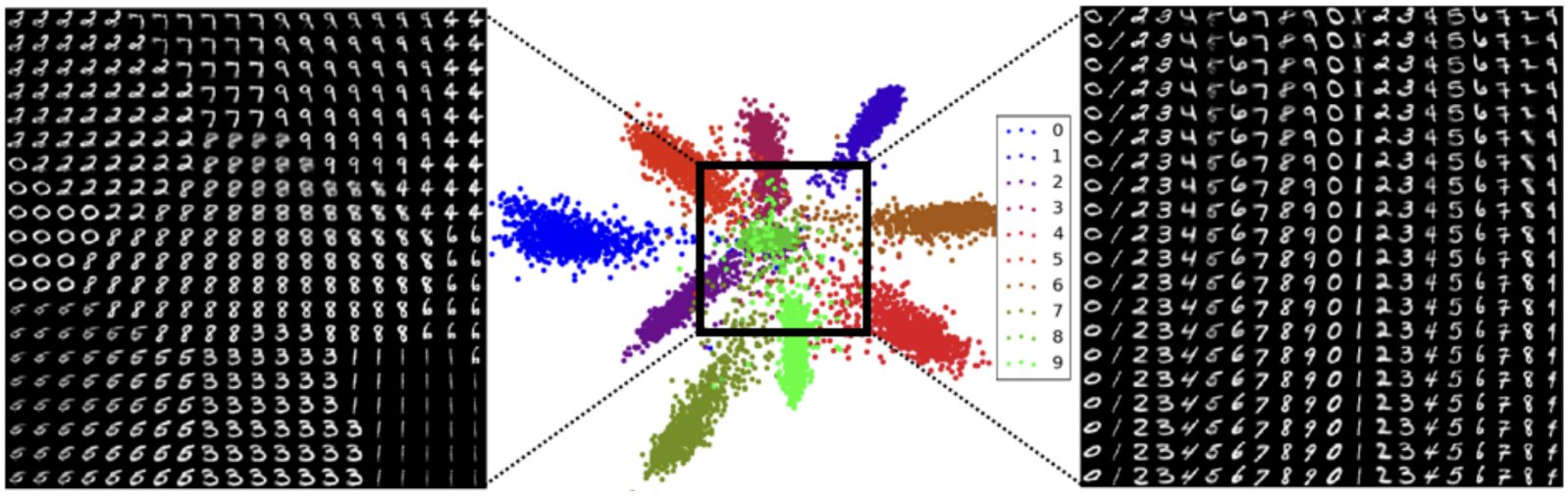



Figure: The latent space (center) clearly distinguishes between different MNIST classes. From a given point in the latent space, we can generate many different images.

# Drug Design & Discovery

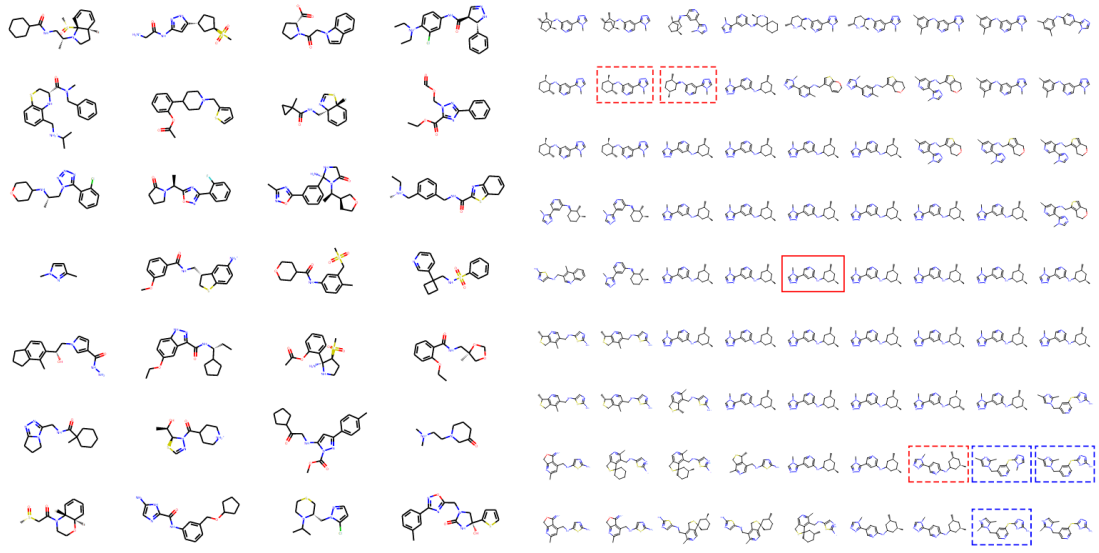


*Figure 6.* **Left**: Random molecules sampled from prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. **Right**: Visualization of the local neighborhood of a molecule in the center. Three molecules highlighted in red dashed box have the same tree structure as the center molecule, but with different graph structure as their clusters are combined differently. The same phenomenon emerges in another group of molecules (blue dashed box).

Figure: We can learn a latent space over molecules, and then sample molecules from different parts of the space. This gives a simple way of comparing molecular structures, which are otherwise complicated objects to try to compare. Image taken from [1]

# Probabilistic Tumor Segmentation

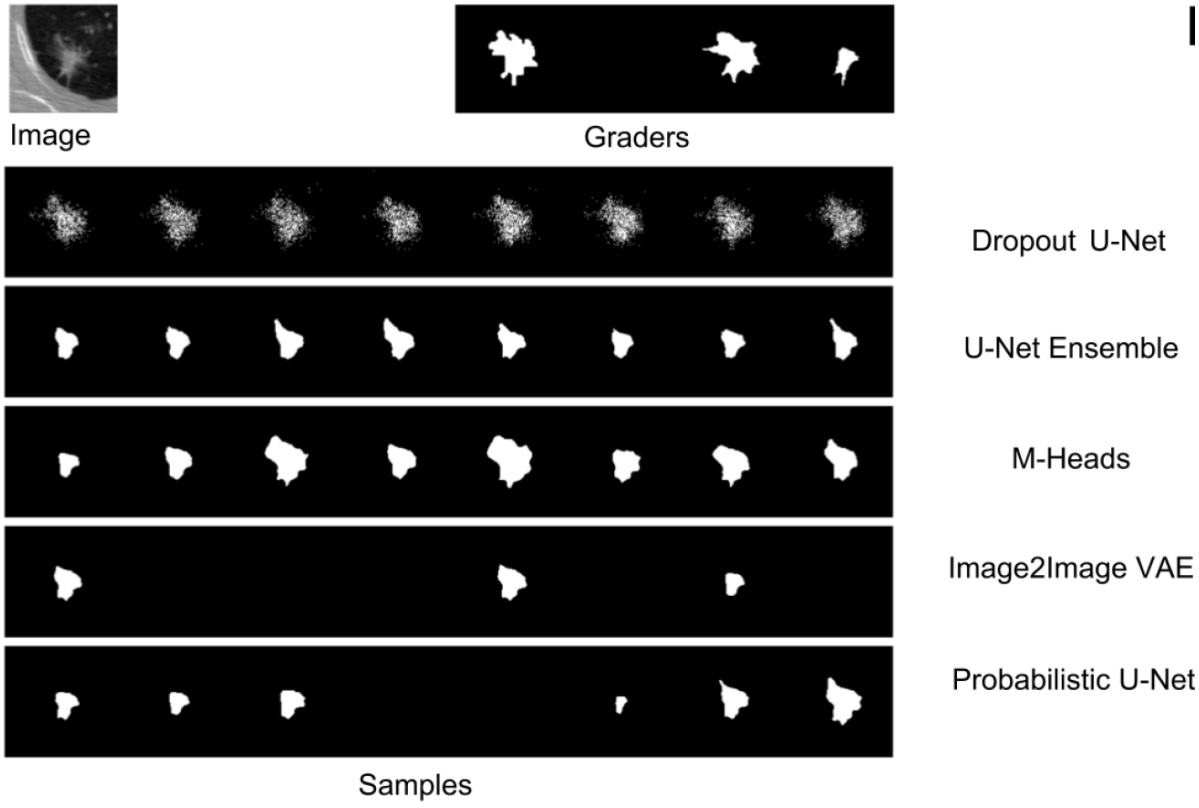



Figure: A probabilistic model over tumor segmentations allows doctors to view several plausible views associated with a single X-ray image. Example taken from [2].

# Conditional Sample

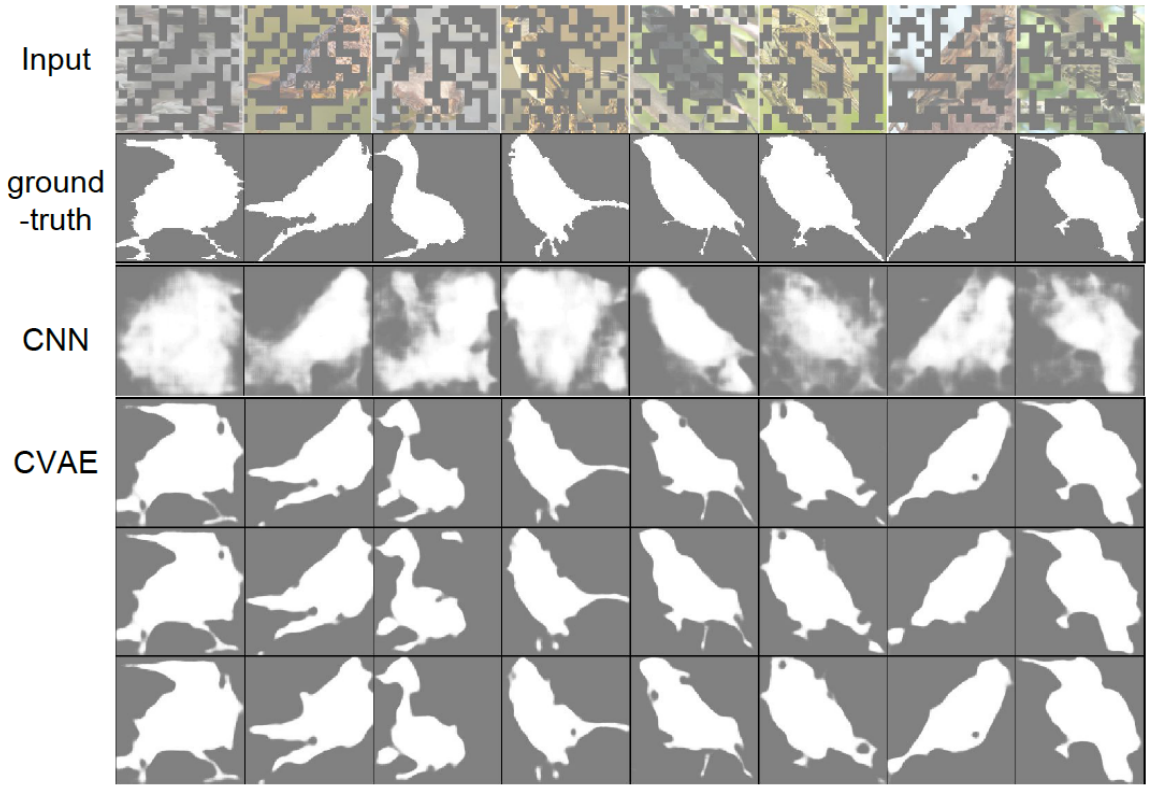



Figure: Since the VAE is a probabilistic model, we can condition on partial information. Here, a segmentation is conditioned on only partially observed input images.

# One-Shot Generalization

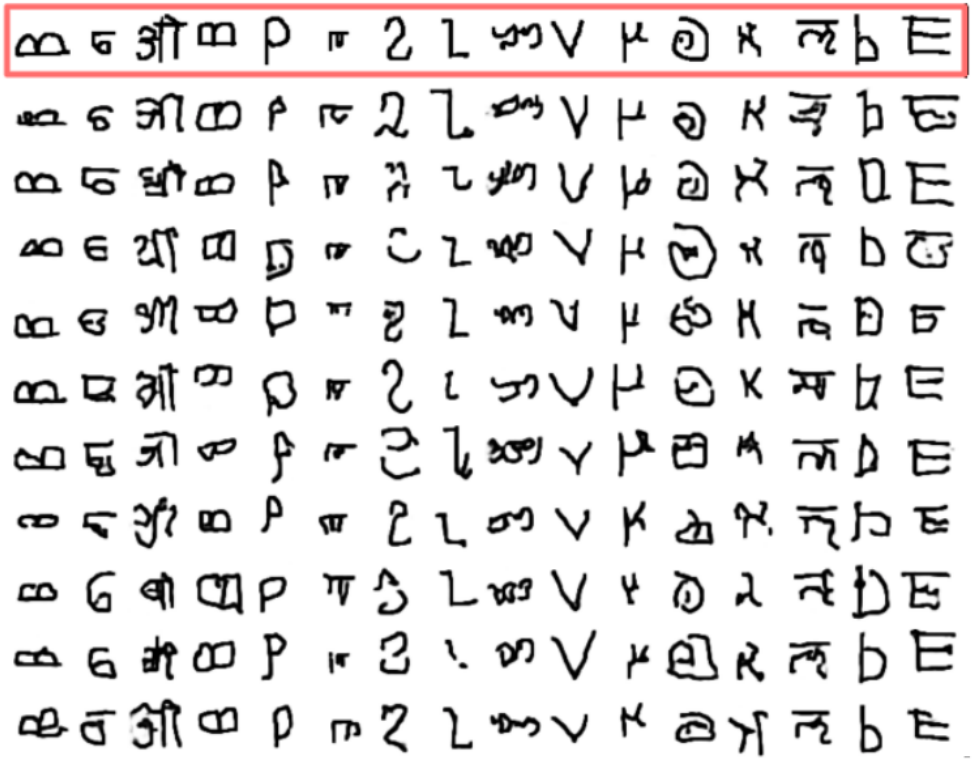


Figure: Given only a few observations from one class, we can start to generate images that look like it, after encoding it properly. Example taken from [4].

# Conclusion

- Latent variables are useful devices for introducing complexity
- To fit latent variable models, need to simultaneously optimize
  - Configuration $z$'s consistent with observed data
  - Global parameters $\theta$ generating data
- Variational Autoencoders approximate full mean-field inference using an inference network with shared parameters
  - Allows application to settings where forwards model is not amenable to analytical formulas

[1] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[2] Simon AA Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus H Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. *arXiv preprint arXiv:1806.05034*, 2018.

[3] Shakir Mohamed. *Generalised Bayesian matrix factorisation models*. PhD thesis, University of Cambridge, 2011.

[4] Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.