

Variational Inference

February 20, 2018

- 1 Generative Models
- 2 Variational Inference
- 3 Stochastic Variational Inference
- 4 Variational Autoencoders

1 Generative Models

2 Variational Inference

3 Stochastic Variational Inference

4 Variational Autoencoders

Generative Models

- Describes how the observed data are generated
- You've seen one example already:
Language modeling : Given x_1, \dots, x_T , fit $p_\theta(x_1, \dots, x_T)$ to the data.

Generative Models: Example

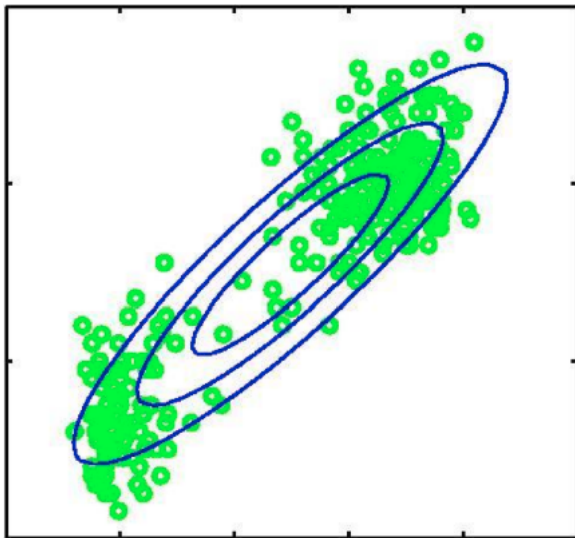
Often helpful to think of the underlying **generative process** (i.e. how one could generate data according to the model). E.g. for a bigram language model,

- 1 Draw $x_1 \sim p(X|Y = \langle s \rangle)$
- 2 Draw $x_2 \sim p(X|Y = x_1)$
- 3 Draw $x_3 \sim p(X|Y = x_2)$
- 4 ...

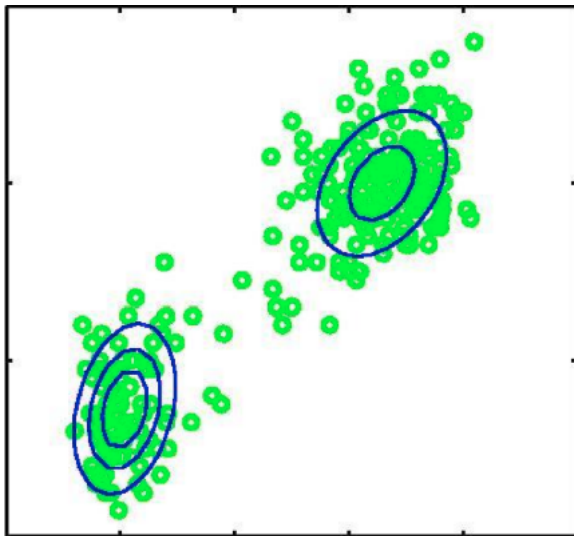
Latent Variable Models

- Language model: no latent variables, factorize $p(\mathbf{x})$ according to chain rule
- Latent variable models: Assumes **observed** data \mathbf{x} are generated from **unobserved** latent variables \mathbf{z}
- Simplest case:
 - 1 Draw \mathbf{z} from prior $p(\mathbf{z})$
 - 2 Draw \mathbf{x} from conditional $p(\mathbf{x}|\mathbf{z})$
- Our setup:
 - 1 $p(\mathbf{z})$ usually simple
 - 2 $p_{\theta}(\mathbf{x}|\mathbf{z})$ parameterized with a deep model θ

Example: Mixture of Gaussians



Example: Mixture of Gaussians



Example: Mixture of Gaussians

Generative process for GMM with K mixture components:

For each data point,

- 1 Sample $z^{(i)} \sim \text{Categorical}(\frac{1}{K})$
- 2 Sample $\mathbf{x}^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}})$

What is the prior? What is θ ? What is $p_{\theta}(\mathbf{x})$? How do we learn?

Example: Deep Generative Models

MNIST digits, $\mathbf{x} \in \mathbb{R}^{784}$

- 1 Sample $\mathbf{z}^{(i)} \sim \mathcal{N}(0, I), \mathbf{z} \in \mathbb{R}^n$
- 2 Feed $\mathbf{z}^{(i)}$ through a one-layer feed-forward network

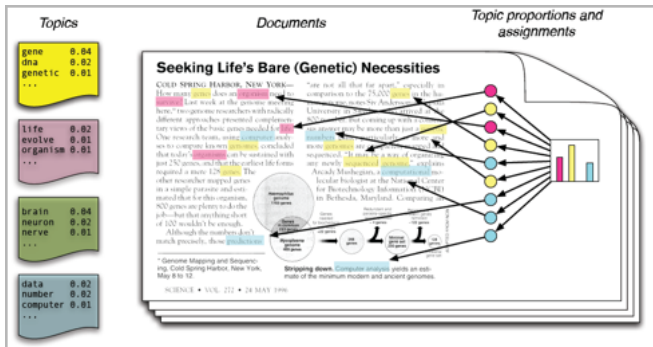
$$\mathbf{h} = \text{ReLU}(\mathbf{W}_1 \mathbf{z})$$

- 3 Sample $\mathbf{x}^{(i)} \sim \sigma(\mathbf{W}_2 \mathbf{h})$

Why Latent Variable Models?

- Learn useful representations from unlabeled data (unsupervised learning)
- Natural way of incorporating multimodality
- Able to capture complex, hierarchical generative processes

Why Latent Variable Models?



Learning in Latent Variable Models

Given data $\mathbf{x}^{(i)}, i = 1, \dots, N$, want to maximize log-likelihood, i.e.

$$\max_{\theta} \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$$

- No latent variables: usual setup
- Latent variables:

$$\max_{\theta} \sum_{i=1}^N \log \int_{\mathbf{z}} p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}^{(i)}) p(\mathbf{z}^{(i)}) d\mathbf{z}$$

(replace \int with sum if \mathbf{z} discrete)

1 Generative Models

2 Variational Inference

3 Stochastic Variational Inference

4 Variational Autoencoders

Learning in Latent Variable Models

Variational Inference:

- Idea: Introduce a **variational** distribution $q_{\lambda}(\mathbf{z})$ parameterized by λ for each data point. Then,

$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\lambda}(\mathbf{z})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] - KL[q_{\lambda}(\mathbf{z})||p(\mathbf{z})]$$

(Derive on board)

- This is called the **evidence lower bound** or **ELBO**
- Running example: Gaussian variational family

$$q_{\lambda}(\mathbf{z}) = \mathcal{N}(\mu, \Sigma)$$

i.e. $\lambda = [\mu, \Sigma]$

Evidence Lower Bound

Learning problem: Find $\lambda^{(i)}, \theta$ that maximize

$$\sum_{i=1}^N \mathbb{E}_{q_{\lambda}^{(i)}(\mathbf{z})} [\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}^{(i)})] - KL[q_{\lambda^{(i)}}(\mathbf{z}) \| p(\mathbf{z}^{(i)})]$$

Coordinate ascent:

- 1 Hold θ fixed, maximize ELBO with respect to $\lambda^{(i)}$'s
- 2 Hold $\lambda^{(i)}$'s fixed, maximize ELBO with respect to θ
- 3 Repeat

In certain cases coordinate ascent admits analytic update formulas

Examining the ELBO

Consider a single point

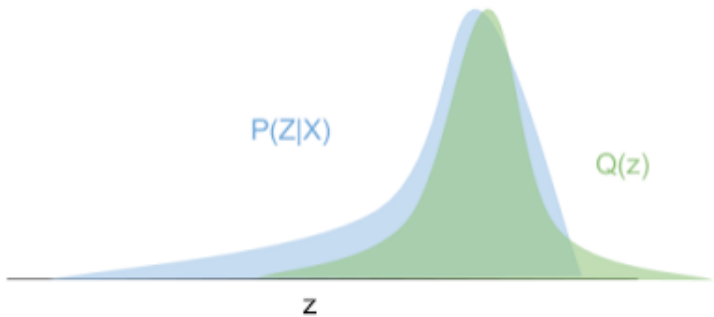
$$ELBO = \log p_{\theta}(\mathbf{x}) - KL[q_{\lambda}(\mathbf{z}) || p_{\theta}(\mathbf{z}|\mathbf{x})]$$

Claim: Setting $q_{\lambda}(\mathbf{z}) = p_{\theta}(\mathbf{z}|\mathbf{x})$ makes the bound tight, i.e.

$$ELBO = \log p_{\theta}(\mathbf{x})$$

- Justifies why optimizing the ELBO is a good idea
- But why can't we do this all the time?

Posterior vs Variational Family



Examining the ELBO

$$ELBO = \log p_{\theta}(\mathbf{x}) - KL[q_{\lambda}(\mathbf{z})||p_{\theta}(\mathbf{z}|\mathbf{x})]$$

- Usually, we will be learning the generative model θ
- But if we are just interested in inference (e.g. at test time), we can minimize with respect to $q_{\lambda}(\mathbf{z})$

$$KL[q_{\lambda}(\mathbf{z})||p_{\theta}(\mathbf{z}|\mathbf{x})]$$

- “Inference as optimization”

Yet another view of ELBO

Rearranging some terms, we also get that

$$\begin{aligned} ELBO &= \log p_{\theta}(\mathbf{x}) - KL[q_{\lambda}(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_{\lambda}(\mathbf{z})}[\log p_{\theta}(\mathbf{x}, \mathbf{z})] + \mathbb{H}[q_{\lambda}(\mathbf{z})] \end{aligned}$$

- 1 Hold θ fixed, set $q_{\lambda}(\mathbf{z}) = p_{\theta}(\mathbf{z}|\mathbf{x})$ (suppose this is tractable)
- 2 Hold $q_{\lambda}(\mathbf{z})$ fixed, maximize

$$\mathbb{E}_{q_{\lambda}(\mathbf{z})}[\log p_{\theta}(\mathbf{x}, \mathbf{z})]$$

($\mathbb{H}[q_{\lambda}(\mathbf{z})]$ constant with respect to θ)

(This is exactly the EM algorithm)

- 1 Generative Models
- 2 Variational Inference
- 3 Stochastic Variational Inference**
- 4 Variational Autoencoders

Stochastic Variational Inference

$$\sum_{i=1}^N \mathbb{E}_{q_{\lambda}^{(i)}(\mathbf{z})} [\log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}^{(i)})] - KL[q_{\lambda^{(i)}}(\mathbf{z}) \| p(\mathbf{z}^{(i)})]$$

- Coordinate ascent is impractical on large datasets (need to find optimal $\lambda^{(i)}$ for all i , then update θ)
- Idea: Just use minibatches (or single datum) \implies Stochastic Variational Inference

Stochastic Variational Inference (Hoffman et al. 2013)

Define ELBO with respect to single datum

$$ELBO(\mathbf{x}, \lambda, \theta) = \mathbb{E}_{q_\lambda(\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL[q_\lambda(\mathbf{z})||p(\mathbf{z})]$$

- 1 Sample $\mathbf{x} \sim p_{\mathcal{D}}(\mathbf{x})$
- 2 Randomly initialize λ_0
- 3 For number of steps, optimize ELBO wrt λ with gradient ascent, i.e. for $k = 1, \dots, K$

$$\lambda_k \leftarrow \lambda_{k-1} + \alpha \nabla_\lambda ELBO(\mathbf{x}, \theta, \lambda_0)$$

- 4 Optimize ELBO wrt θ , i.e.

$$\theta \leftarrow \theta + \alpha \nabla_\theta ELBO(\mathbf{x}, \theta, \lambda_K)$$

Stochastic Variational Inference

Need to compute: $\nabla_{\lambda} ELBO(\mathbf{x}, \theta, \lambda), \nabla_{\theta} ELBO(\mathbf{x}, \theta, \lambda)$

- Easy:

$$\nabla_{\theta} ELBO(\mathbf{x}, \theta, \lambda) = \mathbb{E}_{q_{\lambda}(\mathbf{z})} [\nabla_{\theta} \log p_{\theta}(\mathbf{x}|\mathbf{z})]$$

- Hard:

$$\nabla_{\lambda} ELBO(\mathbf{x}, \theta, \lambda) = \nabla_{\lambda} \mathbb{E}_{q_{\lambda}(\mathbf{z})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] - \nabla_{\lambda} KL[q_{\lambda}(\mathbf{z})||p(\mathbf{z})]$$

In the fully general case, need score function (i.e. policy gradients/
REINFORCE) from last lecture

The Reparameterization Trick

- Until now: $p(\mathbf{z}), q_\lambda(\mathbf{z})$ arbitrary distributions
- From hereon: $p(\mathbf{z}) = N(0, I), q_\lambda(\mathbf{z}) = N(\mu, \Sigma)$
- The variational family is given by Gaussian with mean vector μ and covariances Σ (i.e. $\lambda = [\mu, \Sigma]$)
- Σ usually diagonal
- This allows low-variance estimators for $\nabla_\lambda ELBO(\mathbf{x}, \theta, \lambda)$

The Reparameterization Trick

$$\underbrace{\mathbb{E}_{q_{\lambda}(\mathbf{z})}[\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{1. \text{ Reparameterization trick}} - \underbrace{KL[q_{\lambda}(\mathbf{z})||p(\mathbf{z})]}_{2. \text{ Analytic formula for Gaussian family}}$$

- Reparameterization trick (pathwise derivatives): Exploit the fact that Gaussians are **reparameterizable**
- Drawing \mathbf{z} from $N(\mu, \Sigma)$ is the same as drawing $\epsilon \sim N(0, I)$ and applying the transformation

$$\mathbf{z} = \mu + A\epsilon$$

where $AA^{\top} = \Sigma$ (obtained from, for example, Cholesky decomposition)

- If Σ is diagonal, $\text{diag}(A) = [\sigma_1, \dots, \sigma_n]$

The Reparameterization Trick

$$\lambda = [\mu, \sigma^2]$$

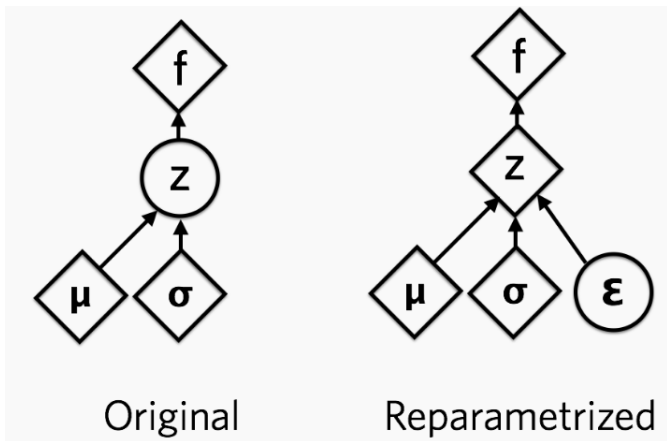
$$\begin{aligned} & \nabla_{\lambda} \mathbb{E}_{q_{\lambda}(\mathbf{z})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] \\ &= \nabla_{\lambda} \mathbb{E}_{\epsilon \sim N(0, I)} [\log p_{\theta}(\mathbf{x}|\mu + \epsilon\sigma)] \\ &= \mathbb{E}_{\epsilon \sim N(0, I)} [\nabla_{\lambda} \log p_{\theta}(\mathbf{x}|\mu + \epsilon\sigma)] \end{aligned}$$

- Now we just need samples from a **fixed** distribution $\epsilon \sim N(0, I)$
- Empirically this has lower variance than REINFORCE estimator

$$\mathbb{E}_{q_{\lambda}(\mathbf{z})} [\log p_{\theta}(\mathbf{x}|\mathbf{z}) \nabla_{\lambda} \log q_{\lambda}(\mathbf{z})]$$

- (But both are unbiased estimators)

The Reparameterization Trick



(Circles are stochastic nodes)

- 1 Generative Models
- 2 Variational Inference
- 3 Stochastic Variational Inference
- 4 Variational Autoencoders

Variational Autoencoders/Amortized Inference

Amortized variational inference (Mnih et al. 2014, Kingma and Welling 2014, Rezende et al. 2014):

- **Predict** the variational parameters to be a function of the input

$$\lambda(\mathbf{x}) = \text{enc}_{\phi}(\mathbf{x})$$

- The **inference network** (or encoder/recognition network), parameterized by ϕ , is shared (i.e. amortized) across all \mathbf{x}
- “Inference as prediction”

Variational Autoencoders/Amortized Inference

Learning problem

$$\max_{\phi, \theta} \mathbb{E}_{q_{enc_{\phi}}(\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] - KL[q_{enc_{\phi}}(\mathbf{z})||p(\mathbf{z})]$$

Why Variational "Autoencoder"

$$\min_{\phi, \theta} \underbrace{\mathbb{E}_{q_{enc_{\phi}}(\mathbf{x})} [-\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction loss}} + \underbrace{KL[q_{enc_{\phi}}(\mathbf{z})||p(\mathbf{z})]}_{\text{Regularizer}}$$

Variational Autoencoders/Amortized Inference

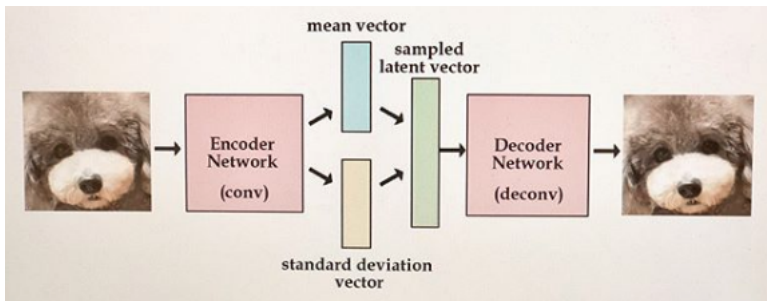
End-to-end training with backprop (no coordinate ascent)

- 1 Sample $\mathbf{x} \sim p_{\mathcal{D}}(\mathbf{x})$
- 2 Run inference network

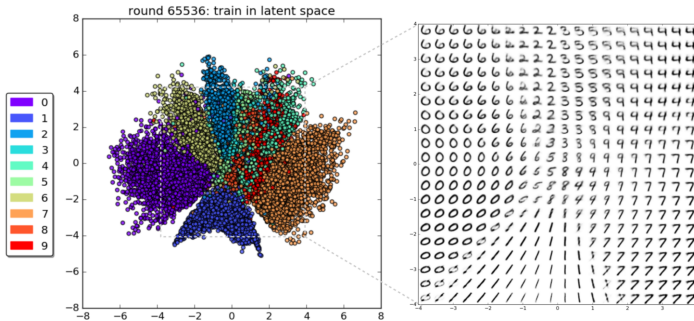
$$\mu(\mathbf{x}), \sigma^2(\mathbf{x}) = \text{enc}_{\phi}(\mathbf{x})$$

- 3 Sample $\epsilon \sim N(0, I)$, reparameterize $\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x})\epsilon$
- 4 Calculate loss $\mathcal{L} = -\log p_{\theta}(\mathbf{x}|\mathbf{z}) + KL(q_{\lambda}(\mathbf{z})\|p(\mathbf{z}))$
- 5 Update θ, ϕ based on $\nabla_{\theta}\mathcal{L}, \nabla_{\phi}\mathcal{L}$

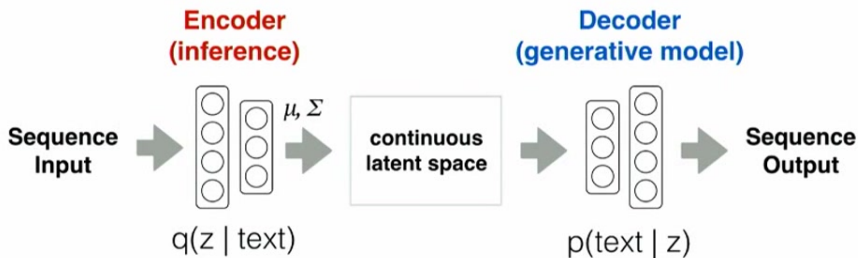
Variational Autoencoders for Images



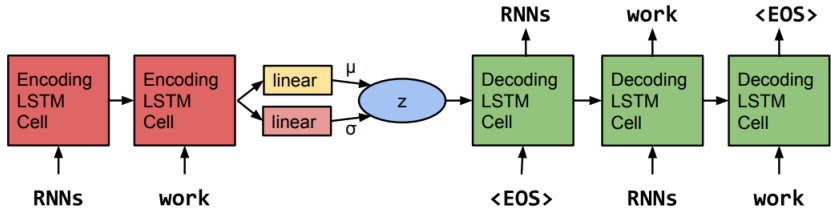
Variational Autoencoders for Images



Variational Autoencoders for Text Processing



Variational Autoencoders for Text Processing



Practical Issues

- Parameterization: Use $\log \sigma^2$ instead of σ^2
- Tricks: KL-annealing, word-dropout

Takeaways

- Variational Inference: “Inference as Optimization”
- Amortized Inference: “Optimization as Prediction”