Machine Learning
Computing with Latent Variables

Rajesh Ranganath

Formal definition

Probabilistic latent variable models:

- Data: $\mathbf{x}$
- Hidden Structure (latent variables): $\mathbf{z}$
- Model: $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ - prior $p(\mathbf{z})$ and likelihood $p(\mathbf{x}|\mathbf{z})$
- Posterior: $p(\mathbf{z}|\mathbf{x})$ - probability of the hidden structure

Astrophysics

[Regier+ 2015]

Astrophysics

Why a latent variable model?

Astrophysics

Prior knowledge about

$$p(\mathbf{x}_i : \text{light-measurement} \,|\, z_i = \text{galaxy}) = \text{Known from physics}$$

Astrophysics

Prior knowledge about

$$p(\mathbf{x}_i : \text{light-measurement} \,|\, z_i = \text{galaxy}) = \text{Known from physics}$$

Posterior over mixture component “classifies”

$$\begin{aligned} & p(z_i = \text{galaxy} \,|\, \mathbf{x}_i) \\ &= \frac{p(z_i = \text{galaxy}) p(\mathbf{x}_i : \text{light-measurement} \,|\, z_i = \text{galaxy})}{\sum_{\text{type} \in \{\text{galaxy, planet,...}\}} p(\mathbf{x}_i : \text{“light''} \,|\, z_i = \text{type}) p(z_i = \text{type})} \end{aligned}$$

Uses of Latent Variable Models

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

Combining Simple Distributions

Take a categorical distribution

$$\text{Categorical}(1 \dots K)$$

Take a normal distribution

$$\text{Normal}(\mu, \sigma)$$

Combine

$$z_i \sim \text{Categorical}(1 \dots K)$$
$$x_i \sim \text{Normal}(\mu_{z_i}, \sigma_{z_i})$$

Get a mixture of Gaussians, which is far more flexible

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

Do Latent Variables Help Predict x

What does it mean to predict **x**?

- In some sense yes

$$p(\mathbf{x}) = p(x^{(1)})p(x^{(2)} \mid x^{(1)})p(x^{(3)} \mid x^{(1)}, x^{(2)})\ldots$$

Predictions

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

How do I know my latent variable is correct? Can I used cross-validation like when we predict Y?

This cannot be checked without assumptions because

- The data generating distribution $F(\mathbf{x})$ is all you get from the data
- $F(\mathbf{x})$ can be modeled without latent variables (though maybe slow)
- For a fixed class, predictions and predictive checks help

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

How can we create graphs?

- Based on prior knowledge
- Based on computational considerations
- Based on the hidden structure useful for a problem

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

Where does one use latent variables? Can we have some more real-world examples?

- We'll have one more in a second

Q + A from Last Year

- Encoding prior knowledge
- Combining simple distributions to create a more complex one
- Uncovering hidden structure

Are latent variables about the noise and getting an accurate data generating distribution?

Yes/No

- Noise variables could be latent variables
- Noise is part of the unpredictability of $\mathbf{x}$. How can the first dimension of $\mathbf{x}$ be predicted?

$$p(\mathbf{x}) = p(x^{(1)})p(x^{(2)}\,|\,x^{(1)})p(x^{(3)}|x^{(1)},x^{(2)})\ldots$$

- Latent variables can exist in concert with noise variables (Gaussian noise)
- Latent variables could also be structure that's unobserved

Uncovering hidden structure: Finding Topics in Documents

Data

- M number of documents
- V number of words
- W: $M \times V$ matrix of words

Hidden Structure

- A group of topics that describe the documents
- Each document contains a distribution over topics

Data

- M number of documents
- V number of words
- W: A $M \times V$ matrix of words

How do we describe the topics?

Data

- M number of documents
- V number of words
- W: A $M \times V$ matrix of words

How do we describe the topics?

A distribution over the words called $\boldsymbol{\beta}_k$

- M number of documents
- V number of words
- W: A $M \times V$ matrix of words

s **How do we describe the document's topic composition?**

A distribution over the topics called $\boldsymbol{\theta}_i$

Have two distributions

- $\boldsymbol{\beta}_k$ distributions over words for each topic
- $\boldsymbol{\theta}_i$ distribution over topics for each document

Priors?

Have two distributions

- $\boldsymbol{\beta}_k$ distributions over words for each topic
- $\boldsymbol{\theta}_i$ distribution over topics for each document

Priors? Dirichlet distribution

Still need a likelihood for data

- M number of documents
- V number of words
- W: A $M \times V$ matrix of words

with hidden structure

- $\boldsymbol{\beta}_k$ distributions over words for each topic
- $\boldsymbol{\theta}_i$ distribution over topics for each document

Assume all documents have same length?

For word m in document l

1. Draw word's topic from $z_{m,l} \sim \text{Categorical}(\boldsymbol{\theta}_i)$
2. Draw topic from for $w_{m,l} \sim \text{Categorical}(\boldsymbol{\beta}_{z_{m,l}})$

Topic Model

For each topic:

1. Draw distribution over words from Dirichlet(α)

For each document:

1. Draw distribution over topics from Dirichlet(κ)

For word m in document l:

1. Draw word's topic from $z_{m,l} \sim \text{Categorical}(\boldsymbol{\theta}_i)$
2. Draw word from topic for $w_{m,l} \sim \text{Categorical}(\boldsymbol{\beta}_{z_{m,l}})$

Topic Model

For each topic:

1. Draw distribution over words from Dirichlet(α)

For each document:

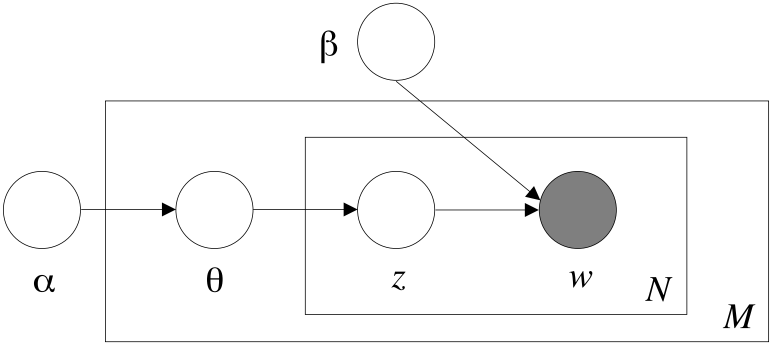
1. Draw distribution over topics from Dirichlet(κ)

For word m in document l:

1. Draw word's topic from $z_{m,l} \sim \text{Categorical}(\boldsymbol{\theta}_i)$
2. Draw word from topic for $w_{m,l} \sim \text{Categorical}(\boldsymbol{\beta}_{z_{m,l}})$

Compute posterior

$$p(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z} \,|\, \mathbf{w}) = \frac{p(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \mathbf{w})}{p(\mathbf{w})}$$


β
α
θ
z
w
N
M

The New York Times

music	book	art	game	show
band	life	museum	knicks	film
songs	novel	show	nets	television
rock	story	exhibition	points	movie
album	books	artist	team	series
jazz	man	artists	season	says
pop	stories	paintings	play	life
song	love	painting	games	man
singer	children	century	night	character
night	family	works	coach	know

theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator	investment	chicken	legislature
directed	house	trading	served	fiscal

Nature

dna	channel	visual	ray	glucose
sequence	channels	stimulus	emission	liver
gene	receptor	subjects	pulsar	enzyme
sequences	voltage	motion	radio	tissue
rna	currents	target	radiation	phosphate
fragment	membrane	stimuli	star	rats
cdna	binding	trials	sources	fraction
mrna	receptors	response	stars	incorporation
genes	neurons	neurons	neutron_star	synthesis
fragments	activation	spatial	pulsars	mgm

war	stars	stars	tube	virus
social	star	observatory	wire	hiv
industrial	disk	the_sun	glass	infection
policy	solar	star	apparatus	disease
economic	galaxy	comet	force	infected
planning	formation	eclipse	heat	aids
men	galaxies	solar	instrument	vaccine
service	galactic	magnitude	electric	viruses
management	massive	photographs	you	viral
labour	objects	planet	iron	host

[Hoffman+ 2013]

Computing the posterior. Is it easy?

Computing the posterior is hard

The posterior distribution

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

The model $p(\mathbf{x}, \mathbf{z})$ is given; the challenge lies in computing $p(\mathbf{x})$

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

This is a high dimensional integral (sum) which in general is (analytically) intractable

Computing the posterior is hard

Bayesian Mixture of Gaussians

$$\begin{aligned} \mu_k &\sim \text{Normal}(0, 1) \\ z_i &\sim \text{Categorical}(1 \ldots K) \\ x_i &\sim \text{Normal}(\mu_{z_i}, 1) \end{aligned}$$

Marginal likelihood is

$$p(\mathbf{x}) = \int \prod_{k=1}^{K} p(\mu_k) \prod_{i=1}^{N} \sum_{j=1}^{K} p(z_i = j) p(x_i \mid \mu_j) d\mu_1 \ldots d\mu_k$$

Swapping

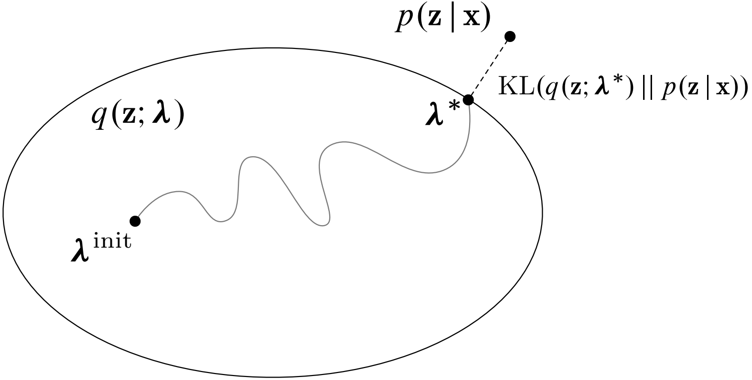
$$p(\mathbf{x}) = \sum_{\mathbf{z}} \prod_{k=1}^{K} \int p(\mu_k) \prod_{i:\mathbf{z}[i]=k}^{N} p(z_i = k) p(x_i \mid \mu_k) d\mu_k$$

Show the equality

$$
\begin{aligned}
p(\mathbf{x}) &= \int \prod_{k=1}^{K} p(\mu_k) \prod_{i=1}^{N} \sum_{j=1}^{K} p(z_i = j) p(x_i \mid \mu_j) d\mu_1 \ldots d\mu_k \\
&= \int \prod_{k=1}^{K} p(\mu_k) \sum_{\mathbf{z}} \prod_{i=1}^{N} p(z_i = z[i]) p(x_i \mid \mu_{z[i]}) d\mu_1 \ldots d\mu_k \\
&= \int \sum_{\mathbf{z}} \prod_{k=1}^{K} p(\mu_k) \prod_{i=1}^{N} p(z_i = z[i]) p(x_i \mid \mu_{z[i]}) d\mu_1 \ldots d\mu_k \\
&= \sum_{\mathbf{z}} \int \prod_{k=1}^{K} p(\mu_k) \prod_{i=1}^{N} p(z_i = z[i]) p(x_i \mid \mu_{z[i]}) d\mu_1 \ldots d\mu_k \\
&= \sum_{\mathbf{z}} \prod_{k=1}^{K} \int p(\mu_k) \prod_{i:\mathbf{z}[i]=k} p(z_i = k) p(x_i \mid \mu_k) d\mu_k
\end{aligned}
$$

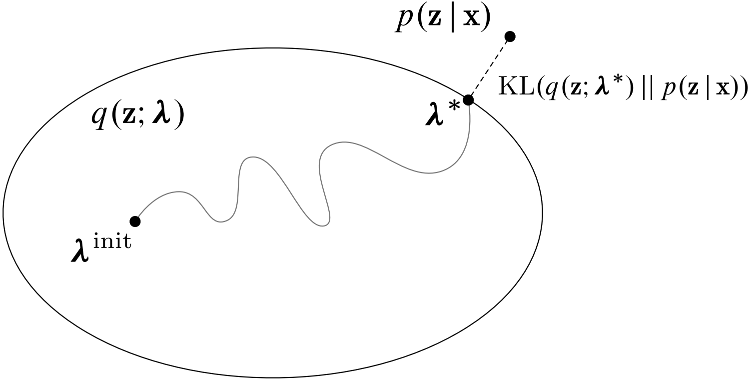
Uses $\int_{a,b} f(a)g(b) = \int_a f(a) \int_b g(b)$

Variational Inference



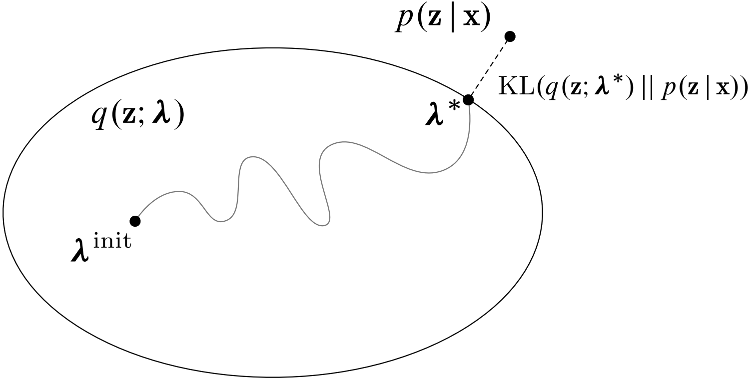
Posit a family of distributions $q(\mathbf{z}; \boldsymbol{\lambda})$ indexed with parameter $\boldsymbol{\lambda}$

Variational Inference



Find $\boldsymbol{\lambda}$ such that q is close to $p(\mathbf{z} \mid \mathbf{x})$

Variational Inference



Closeness measured by the KL divergence

$$
\begin{aligned}
KL(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z}\,|\,\mathbf{x})) &= \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda}) - \log p(\mathbf{z}\,|\,\mathbf{x})] \\
&= \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda}) - \log p(\mathbf{z},\mathbf{x}) + \log p(\mathbf{x})] \\
&= \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\lambda}) - \log p(\mathbf{z},\mathbf{x})] + \log p(\mathbf{x})
\end{aligned}
$$

Equivalently,

$$
\begin{aligned}
\log p(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})] + KL(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z}\,|\,\mathbf{x})) \\
&\geq \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})]
\end{aligned}
$$

A lower bound on the evidence $\log p(\mathbf{x})$

The Evidence Lower Bound

$$\mathscr{L}(\boldsymbol{\lambda}) = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z}; \boldsymbol{\lambda})]$$

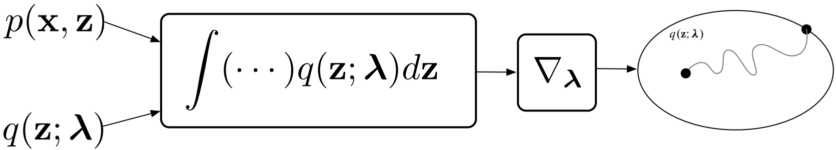
- KL is intractable; VI optimizes the **evidence lower bound** (ELBO)
  - It is a lower bound on $\log p(\mathbf{x})$
  - Maximizing the ELBO is equivalent to minimizing the KL
- The ELBO trades off two terms
  - The first term prefers $q(\cdot)$ to place its mass on the MAP estimate
  - The second term encourages $q(\cdot)$ to be diffuse
- Approximation q is chosen to match types

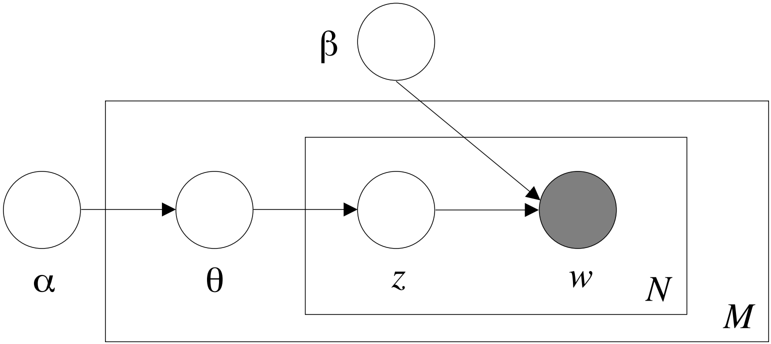
The Evidence Lower Bound

$$\mathscr{L}(\boldsymbol{\lambda}) = \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}(q(\mathbf{z}; \boldsymbol{\lambda}) || p(\mathbf{z}))$$

- KL is intractable; VI optimizes the **evidence lower bound** (ELBO)
  - It is a lower bound on $\log p(\mathbf{x})$
  - Maximizing the ELBO is equivalent to minimizing the KL
- The ELBO trades off two terms
  - The first term prefers $q(\cdot)$ to maximize the likelihood
  - The second term regularizes $q(\cdot)$ to the prior
- Approximation q is chosen to match types

The Recipe


$p(\mathbf{x}, \mathbf{z})$
$q(\mathbf{z}; \boldsymbol{\lambda})$
$\int (\cdots) q(\mathbf{z}; \boldsymbol{\lambda}) d\mathbf{z}$
$\nabla_{\boldsymbol{\lambda}}$
$q(\mathbf{z}; \boldsymbol{\lambda})$


β
α
θ
z
w
N
M

VI for LDA

A.1 Computing $\mathbf{E}[\log(\theta_i \mid \alpha)]$

The need to compute the expected value of the log of a single probability component under the Dirichlet arises repeatedly in deriving the inference and parameter estimation procedures for LDA. This value can be easily computed from the natural parameterization of the exponential family representation of the Dirichlet distribution.

Recall that a distribution is in the exponential family if it can be written in the form:

$$p(x \mid \eta) = h(x)\exp\left\{\eta^T T(x) - A(\eta)\right\},$$

where η is the natural parameter, $T(x)$ is the sufficient statistic, and $A(\eta)$ is the log of the normalization factor.

We can write the Dirichlet in this form by exponentiating the log of Eq. (1):

$$p(\theta \mid \alpha) = \exp\left\{\left(\sum_{i=1}^k(\alpha_i - 1)\log\theta_i\right) + \log\Gamma\left(\sum_{i=1}^k \alpha_i\right) - \sum_{i=1}^k \log\Gamma(\alpha_i)\right\}.$$

From this form, we immediately see that the natural parameter of the Dirichlet is $\eta_i = \alpha_i - 1$ and the sufficient statistic is $T(\theta_i) = \log\theta_i$. Furthermore, using the general fact that the derivative of the log normalization factor with respect to the natural parameter is equal to the expectation of the sufficient statistic, we obtain:

$$\mathrm{E}[\log\theta_i \mid \alpha] = \Psi(\alpha_i) - \Psi\left(\sum_{j=1}^k \alpha_j\right)$$

where Ψ is the digamma function, the first derivative of the log Gamma function.

A.3.2 Variational Dirichlet

Next, we maximize Eq. (15) with respect to γ_i, the ith component of the posterior Dirichlet parameter. The terms containing γ_i are:

$$\begin{aligned} L_{[\gamma]} &= \sum_{i=1}^k (\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) + \sum_{n=1}^N \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &\quad - \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \log\Gamma(\gamma_i) - \sum_{i=1}^k (\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right). \end{aligned}$$

This simplifies to:

$$L_{[\gamma]} = \sum_{i=1}^k \left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right)\left(\alpha_i + \sum_{n=1}^N \phi_{ni} - \gamma_i\right) - \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \log\Gamma(\gamma_i).$$

We take the derivative with respect to γ_i:

$$\frac{\partial L}{\partial \gamma_i} = \Psi'(\gamma_i)\left(\alpha_i + \sum_{n=1}^N \phi_{ni} - \gamma_i\right) - \Psi'\left(\sum_{j=1}^k \gamma_j\right)\sum_{j=1}^k\left(\alpha_j + \sum_{n=1}^N \phi_{nj} - \gamma_j\right).$$

Setting this equation to zero yields a maximum at:

$$\gamma_i = \alpha_i + \sum_{n=1}^N \phi_{ni}. \tag{17}$$

Since Eq. (17) depends on the variational multinomial ϕ, full variational inference requires alternating between Eqs. (16) and (17) until the bound converges.

Finally, we expand Eq. (14) in terms of the model parameters (α, β) and the variational parameters (γ, ϕ). Each of the five lines below expands one of the five terms in the bound:

$$\begin{aligned} L(\gamma, \phi; \alpha, \beta) &= \log\Gamma\left(\sum_{j=1}^k \alpha_j\right) - \sum_{i=1}^k \log\Gamma(\alpha_i) + \sum_{i=1}^k (\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &\quad + \sum_{n=1}^N\sum_{i=1}^k \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &\quad + \sum_{n=1}^N\sum_{i=1}^k\sum_{j=1}^V \phi_{ni} w_n^j \log\beta_{ij} \\ &\quad - \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \sum_{i=1}^k \log\Gamma(\gamma_i) - \sum_{i=1}^k (\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &\quad - \sum_{n=1}^N\sum_{i=1}^k \phi_{ni}\log\phi_{ni}, \end{aligned} \tag{15}$$

where we have made use of Eq. (8).

In the following two sections, we show how to maximize this lower bound with respect to the variational parameters ϕ and γ.

A.3.1 Variational multinomial

We first maximize Eq. (15) with respect to ϕ_{ni}, the probability that the nth word is generated by latent topic i. Observe that this is a constrained maximization since $\sum_{i=1}^k \phi_{ni} = 1$.

We form the Lagrangian by isolating the terms which contain ϕ_{ni} and adding the appropriate Lagrange multipliers. Let β_{iv} be $p(w_n^v = 1 \mid z^i = 1)$ for the appropriate v. (Recall that each w_n is a vector of size V with exactly one component equal to one; we can select the unique v such that $w_n^v = 1$):

$$L_{[\phi_{ni}]} = \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) + \phi_{ni}\log\beta_{iv} - \phi_{ni}\log\phi_{ni} + \lambda_n\left(\sum_{j=1}^k \phi_{ni} - 1\right),$$

where we have dropped the arguments of L for simplicity, and where the subscript ϕ_{ni} denotes that we have retained only those terms in L that are a function of ϕ_{ni}. Taking derivatives with respect to ϕ_{ni}, we obtain:

$$\frac{\partial L}{\partial \phi_{ni}} = \Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right) + \log\beta_{iv} - \log\phi_{ni} - 1 + \lambda.$$

Setting this derivative to zero yields the maximizing value of the variational parameter ϕ_{ni} (cf. Eq. 6):

$$\phi_{ni} \propto \beta_{iv}\exp\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right). \tag{16}$$

The Variational Inference Recipe

Start with a model:

$$p(\mathbf{z}, \mathbf{x})$$

The Variational Inference Recipe

Choose a variational approximation:

$$q(\mathbf{z}; \boldsymbol{\lambda})$$

The Variational Inference Recipe

Write down the ELBO:

$$\mathscr{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\lambda})]$$

The Variational Inference Recipe

Compute the expectation(integral):

$$\text{Example: } \mathscr{L}(\boldsymbol{\lambda}) = x\boldsymbol{\lambda}^2 + \log \boldsymbol{\lambda}$$

The Variational Inference Recipe

Take derivatives:

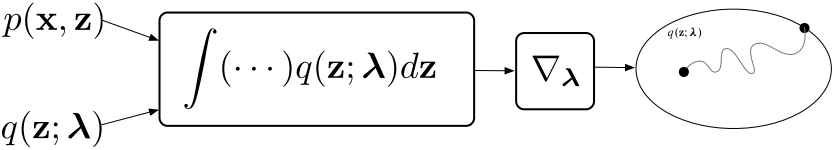
$$\text{Example: } \nabla_{\lambda}\mathscr{L}(\boldsymbol{\lambda}) = 2x\boldsymbol{\lambda} + \frac{1}{\boldsymbol{\lambda}}$$

The Variational Inference Recipe

Optimize:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho_t \nabla_{\boldsymbol{\lambda}} \mathscr{L}$$

The Variational Inference Recipe



Example: Bayesian Logistic Regression

- Data pairs y_i, x_i
- x_i are covariates
- y_i are label
- z is the regression coefficient
- Generative process

$$p(z) \sim N(0,1)$$
$$p(y_i \mid x_i, z) \sim \text{Bernoulli}(\sigma(zx_i))$$

VI for Bayesian Logistic Regression

Assume:

- We have one data point (y, x)
- The approximating family q is the normal; $\boldsymbol{\lambda} = (\mu, \sigma^2)$

The ELBO is

$$\mathscr{L}(\mu, \sigma^2) = \mathbb{E}_q[\log p(z) + \log p(y \,|\, x, z) - \log q(z)]$$

VI for Bayesian Logistic Regression

$$
\begin{aligned}
&\mathscr{L}(\mu, \sigma^2) \\
&\quad = \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)]
\end{aligned}
$$

VI for Bayesian Logistic Regression

$$
\begin{aligned}
&\mathcal{L}(\mu, \sigma^2) \\
&\quad = \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&\quad = \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C
\end{aligned}
$$

VI for Bayesian Logistic Regression

$$
\begin{aligned}
&\mathscr{L}(\mu, \sigma^2) \\
&= \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&= \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[yxz - \log(1 + \ exp(xz))]
\end{aligned}
$$

VI for Bayesian Logistic Regression

$$
\begin{aligned}
&\mathcal{L}(\mu, \sigma^2) \\
&\quad = \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&\quad = \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log\sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&\quad = \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log\sigma^2 + \mathbb{E}_q[yxz - \log(1 + \ exp(xz))] \\
&\quad = \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log\sigma^2 + yx\mu - \mathbb{E}_q[\log(1 + \exp(xz))]
\end{aligned}
$$

VI for Bayesian Logistic Regression

$$
\begin{aligned}
&\mathscr{L}(\mu, \sigma^2) \\
&= \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&= \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[yxz - \log(1 + \ exp(xz))] \\
&= \quad -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + yx\mu - \mathbb{E}_q[\log(1 + \exp(xz))]
\end{aligned}
$$

We are stuck.

1. We cannot analytically take that expectation.
2. The expectation hides the objectives dependence on the variational parameters. This makes it hard to directly optimize.

Options?

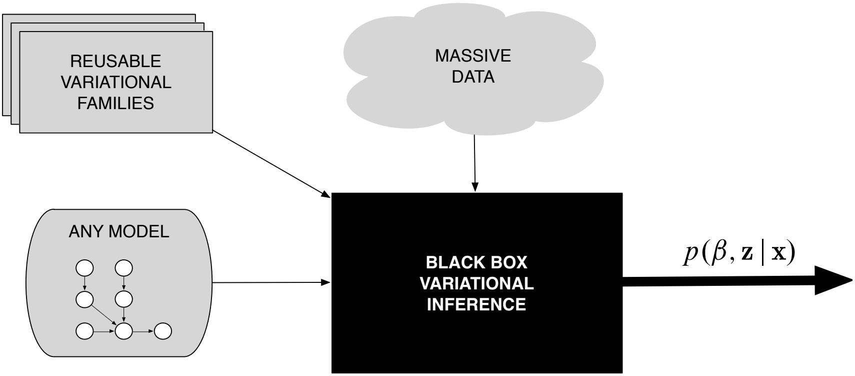
- Derive a model specific bound:
  [Jordan and Jaakola; 1996], [Braun and McAuliffe; 2008], others
- More general approximations that require model-specific analysis:
  [Wang and Blei; 2013], [Knowles and Minka; 2011]

Nonconjugate Models

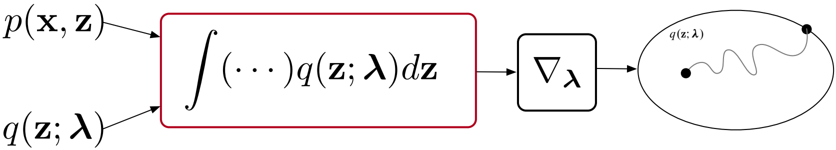
- Nonlinear Time series Models
- Deep Latent Gaussian Models
- Models with Attention (such as DRAW)
- Generalized Linear Models (Poisson Regression)
- Stochastic Volatility Models
- Discrete Choice Models
- Bayesian Neural Networks
- Deep Exponential Families (e.g. Sparse Gamma or Poisson)
- Correlated Topic Model (including nonparametric variants)
- Sigmoid Belief Network

We need a solution that does not entail model specific work

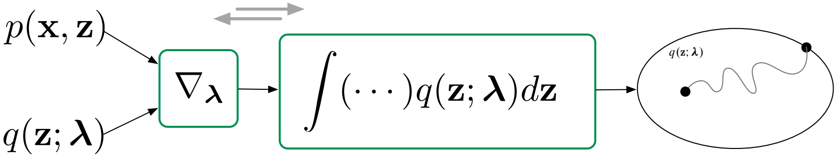
Black Box Variational Inference (BBVI)



The Problem in the Classical VI Recipe



The New VI Recipe



Use stochastic optimization!

Computing Gradients of Expectations

- Define

$$g(\mathbf{z}, \boldsymbol{\lambda}) = \log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})$$

- What is $\nabla_{\boldsymbol{\lambda}} \mathscr{L}$

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \mathscr{L} &= \nabla_{\boldsymbol{\lambda}} \int q(\mathbf{z}; \boldsymbol{\lambda}) g(\mathbf{z}, \boldsymbol{\lambda}) d\mathbf{z} \\
&= \int \nabla_{\boldsymbol{\lambda}} q(\mathbf{z}; \boldsymbol{\lambda}) g(\mathbf{z}, \boldsymbol{\lambda}) + q(\mathbf{z}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} g(\mathbf{z}, \boldsymbol{\lambda}) d\mathbf{z} \\
&= \int q(\mathbf{z}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda}) g(\mathbf{z}, \boldsymbol{\lambda}) + q(\mathbf{z}; \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} g(\mathbf{z}, \boldsymbol{\lambda}) d\mathbf{z} \\
&= \mathbb{E}_{q(\mathbf{z}; \boldsymbol{\lambda})} [\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda}) g(\mathbf{z}, \boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} g(\mathbf{z}, \boldsymbol{\lambda})]
\end{aligned}$$

Using $\nabla_{\boldsymbol{\lambda}} \log q = \frac{\nabla_{\boldsymbol{\lambda}} q}{q}$

Roadmap

- **Score Function Gradients**
- **Reparameterization Gradients**

Score Function Gradients of the ELBO

Score Function Estimator

$$\mathscr{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\lambda})]$$

Recall

$$\nabla_{\boldsymbol{\lambda}}\mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z};\boldsymbol{\lambda}) g(\mathbf{z},\boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} g(z,\boldsymbol{\lambda})]$$

Simplify:

$$\mathbb{E}_q[\nabla_{\boldsymbol{\lambda}} g(\mathbf{z},\boldsymbol{\lambda})] = \mathbb{E}_q[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z};\boldsymbol{\lambda})] = 0$$

Gives the gradient:

$$\nabla_{\boldsymbol{\lambda}}\mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z};\boldsymbol{\lambda})(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\lambda}))]$$

Sometimes called likelihood ratio or REINFORCE gradients

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

Noisy Unbiased Gradients

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

Noisy unbiased gradients with Monte Carlo!

$$\frac{1}{S}\sum_{s=1}^{S} \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}_s;\boldsymbol{\lambda})(\log p(\mathbf{x},\mathbf{z}_s) - \log q(\mathbf{z}_s;\boldsymbol{\lambda})),$$

where $\mathbf{z}_s \sim q(\mathbf{z};\boldsymbol{\lambda})$

Basic BBVI

Algorithm 1: Basic Black Box Variational Inference

Input : Model $\log p(\mathbf{x}, \mathbf{z})$,
Variational approximation $q(\mathbf{z}; \boldsymbol{\lambda})$

Output : Variational Parameters: $\boldsymbol{\lambda}$

while *not converged* **do**

$\mathbf{z}[s] \sim q$ // **Draw** S **samples from** q

$\rho = t$-th value of a Robbins Monro sequence

$\boldsymbol{\lambda} = \boldsymbol{\lambda} + \rho \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}[s]; \boldsymbol{\lambda})(\log p(\mathbf{x}, \mathbf{z}[s]) - \log q(\mathbf{z}[s]; \boldsymbol{\lambda}))$

$t = t + 1$

end

The requirements for inference

The noisy gradient:

$$\frac{1}{S}\sum_{s=1}^{S}\nabla_{\boldsymbol{\lambda}}\log q(\mathbf{z}_s;\boldsymbol{\lambda})(\log p(\mathbf{x},\mathbf{z}_s)-\log q(\mathbf{z}_s;\boldsymbol{\lambda})),$$

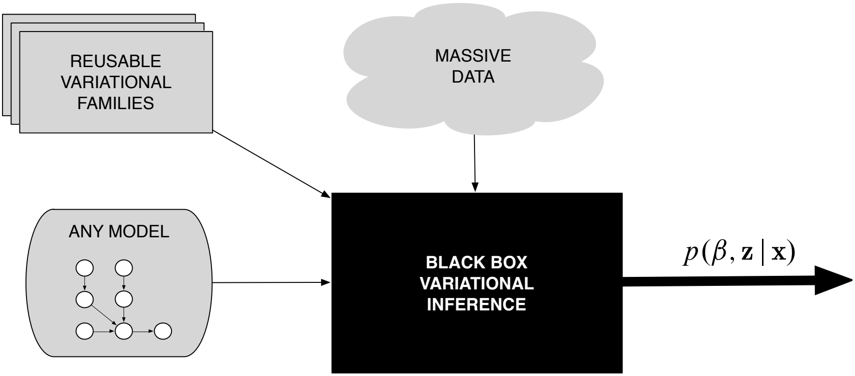
$$\text{where } \mathbf{z}_s \sim q(\mathbf{z};\boldsymbol{\lambda})$$

To compute the noisy gradient of the ELBO we need

- Sampling from $q(\mathbf{z})$
- Evaluating $\nabla_{\boldsymbol{\lambda}}\log q(\mathbf{z};\boldsymbol{\lambda})$
- Evaluating $\log p(\mathbf{x},\mathbf{z})$ and $\log q(\mathbf{z})$

There is no model specific work: black box criteria are satisfied

Black Box Variational Inference



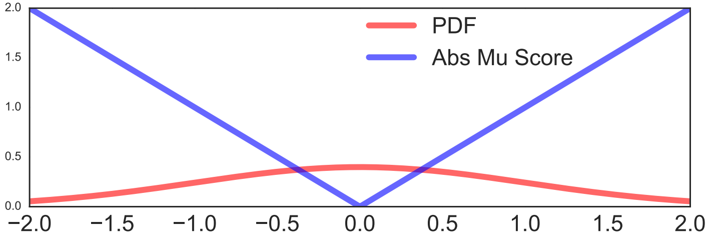
Discussion

Variational Inference for Bayesian Mixtures of Gaussians

Problem: Basic BBVI doesn't work

Variance of the gradient can be a problem

$$\mathrm{Var}_{q(\mathbf{z};\nu)} = \mathbb{E}_{q(\mathbf{z};\nu)}[(\nabla_\nu \log q(\mathbf{z};\nu)(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)) - \nabla_\nu \mathscr{L})^2].$$

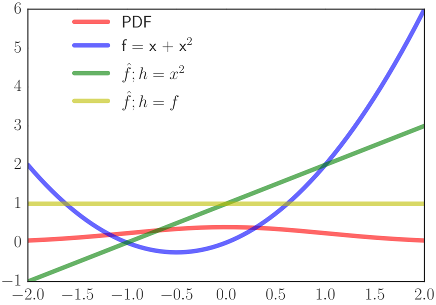


Intuition:
Sampling rare values can lead to large scores and thus high variance

Solution: Control Variates

Replace with f with $\hat{f}$ where $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$. General such class:

$$\hat{f}(z) \triangleq f(z) - a(h(z) - \mathbb{E}[h(z)])$$

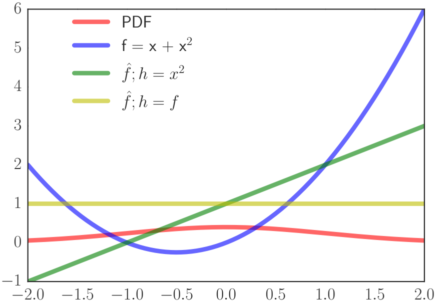


- h is a function of our choice
- a is chosen to minimize the variance
- Good h have high correlation with the original function f

Solution: Control Variates

Replace with f with $\hat{f}$ where $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$. General such class:

$$\hat{f}(z) \triangleq f(z) - a(h(z) - \mathbb{E}[h(z)])$$

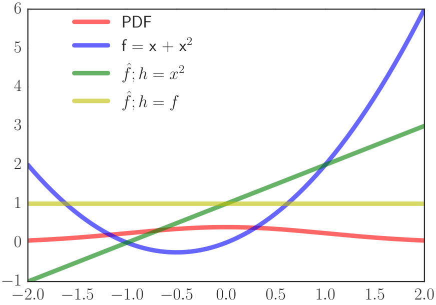


- For variational inference we need functions with known q expectation
- Set h as $\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda})$
- Simple as $\mathbb{E}_q[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda})] = 0$ for any q

Solution: Control Variates

Replace with f with $\hat{f}$ where $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$. General such class:

$$\hat{f}(z) \triangleq f(z) - a(h(z) - \mathbb{E}[h(z)])$$



Many of the other techniques from Monte Carlo can help:

- *Importance Sampling, Quasi Monte Carlo, Rao-Blackwellization*

[Ruiz+ 2016; Ranganath+2014; Titsias+2015; Mnih+2016]

Nonconjugate Models

- Nonlinear Time series Models
- Deep Latent Gaussian Models
- Models with Attention (such as DRAW)
- Generalized Linear Models (Poisson Regression)
- Stochastic Volatility Models
- Discrete Choice Models
- Bayesian Neural Networks
- Deep Exponential Families (e.g. Sparse Gamma or Poisson)
- Correlated Topic Model (including nonparametric variants)
- Sigmoid Belief Network

We can design models based on data rather than inference.

More Assumptions?

The current black box criteria

- Sampling from $q(\mathbf{z})$
- Evaluating $\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z}; \boldsymbol{\lambda})$
- Evaluating $\log p(\mathbf{x}, \mathbf{z})$ and $\log q(\mathbf{z})$

Can we make additional assumptions that are not too restrictive?

Pathwise Gradients of the ELBO

Reparameterization Estimator

Assume

1. $\mathbf{z} = t(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$ for $\boldsymbol{\epsilon} \sim s(\boldsymbol{\epsilon})$ implies $\mathbf{z} \sim q(\mathbf{z}; \boldsymbol{\lambda})$
   Example:

$$\begin{aligned}
\epsilon &\sim \text{Normal}(0, 1) \\
z &= \epsilon\sigma + \mu \\
&\rightarrow z \sim \text{Normal}(\mu, \sigma^2)
\end{aligned}$$

2. $\log p(\mathbf{x}, \mathbf{z})$ and $\log q(\mathbf{z})$ are differentiable with respect to $\mathbf{z}$

Reparameterization Estimator

Recall

$$\nabla_{\boldsymbol{\lambda}} \mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z};\boldsymbol{\lambda}) g(\mathbf{z},\boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} g(z,\boldsymbol{\lambda})]$$

Rewrite using using $\mathbf{z} = t(\boldsymbol{\epsilon},\boldsymbol{\lambda})$

$$\nabla_{\boldsymbol{\lambda}} \mathscr{L} = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\lambda}} \log s(\boldsymbol{\epsilon}) g(t(\boldsymbol{\epsilon},\boldsymbol{\lambda}),\boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} g(t(\boldsymbol{\epsilon},\boldsymbol{\lambda}),\boldsymbol{\lambda})]$$

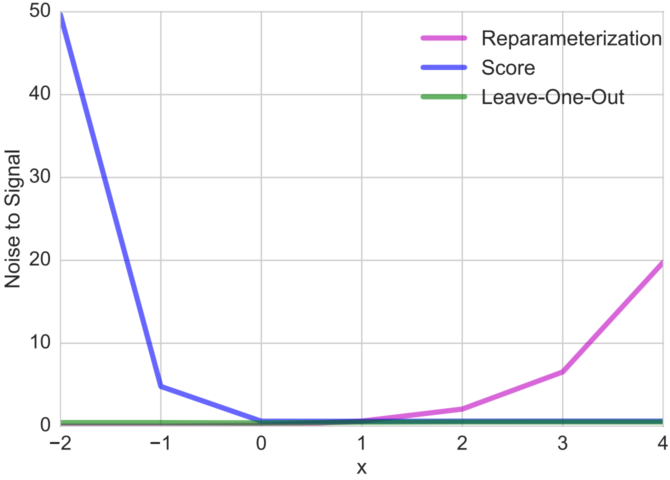
To differentiate:

$$\begin{aligned}
\nabla \mathscr{L}(\boldsymbol{\lambda}) &= \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\lambda}} g(t(\boldsymbol{\epsilon},\boldsymbol{\lambda}),\boldsymbol{\lambda})] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\lambda})] \nabla_{\boldsymbol{\lambda}} t(\boldsymbol{\epsilon},\boldsymbol{\lambda}) - \nabla_{\boldsymbol{\lambda}} \log q(\mathbf{z};\boldsymbol{\lambda})] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\lambda})] \nabla_{\boldsymbol{\lambda}} t(\boldsymbol{\epsilon},\boldsymbol{\lambda})]
\end{aligned}$$

This is also known as the pathwise gradient.

[Glasserman 1991; Fu 2006; Kingma+ 2014; Rezende+ 2014; Titsias+ 2014]

Variance Comparison



[R+ 2018]

What’s an example problem that might have gradients where one is better than the other?

Score Function Estimator vs. Reparameterization Estimator

Score Function

- Differentiates the density $\nabla_{\boldsymbol{\lambda}} q(z; \boldsymbol{\lambda})$
- Works for discrete and continuous models
- Works for large class of variational approximations
- Variance can be a big problem

Pathwise

- Differentiates the function $\nabla_z [\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\lambda})]$
- Requires differentiable models
- Requires variational approximation to have form $\mathbf{z} = t(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$
- Generally better behaved variance

How do we use both estimators at the same time?

Parameter Learning with Variational Inference

Train model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ by maximum likelihood

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) dz$$

Hard to compute the integral. If posterior was known,

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})}$$

Posterior is hard because of integration (unknown $p(\mathbf{x})$).

Parameter Learning with Variational Inference

Maximize lower bound on the likelihood

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + KL(q(\mathbf{z}; \boldsymbol{\lambda}) || p(\mathbf{z} \mid \mathbf{x})) \\ &\geq \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] := \mathscr{L}(\boldsymbol{\lambda}, \boldsymbol{\theta})\end{aligned}$$

A lower bound on the evidence $\log p_{\boldsymbol{\theta}}(\mathbf{x})$

Parameter Learning with Variational Inference

Maximize lower bound on the likelihood

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})] + KL(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z}\,|\,\mathbf{x})) \\ &\geq \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})] := \mathscr{L}(\boldsymbol{\lambda},\boldsymbol{\theta})\end{aligned}$$

A lower bound on the evidence $\log p_{\boldsymbol{\theta}}(\mathbf{x})$

Optimize $\mathscr{L}(\boldsymbol{\lambda},\boldsymbol{\theta})$ using gradients

- Use standard gradients for $\boldsymbol{\theta}$
- Use score/reparameterization gradients for $\boldsymbol{\lambda}$

Parameter Learning with Variational Inference

Maximize lower bound on the likelihood

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})] + KL(q(\mathbf{z};\boldsymbol{\lambda})||p(\mathbf{z}\,|\,\mathbf{x})) \\ &\geq \mathbb{E}_q[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z};\boldsymbol{\lambda})] := \mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\theta})\end{aligned}$$

A lower bound on the evidence $\log p_{\boldsymbol{\theta}}(\mathbf{x})$

Could instead maximize $\boldsymbol{\theta}$ with $\boldsymbol{\lambda}$ fixed and vice-versa

- $\boldsymbol{\lambda}_t = \arg\max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\theta}_{t-1})$
- $\boldsymbol{\theta}_t = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\lambda}_t,\boldsymbol{\theta})$

Called coordinate ascent

Parameter Learning with Variational Inference

Maximize lower bound on the likelihood

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] + KL(q(\mathbf{z}; \boldsymbol{\lambda}) || p(\mathbf{z} \mid \mathbf{x})) \\
&\geq \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}; \boldsymbol{\lambda})] := \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\theta})
\end{aligned}
$$

A lower bound on the evidence $\log p_{\boldsymbol{\theta}}(\mathbf{x})$

Could instead maximize $\boldsymbol{\theta}$ and choose optimal $q = p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$

- Compute optimal $q_t = p_{\boldsymbol{\theta}_{t-1}}(\mathbf{z} \mid \mathbf{x})$
- $\boldsymbol{\theta}_t = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q_t, \boldsymbol{\theta})$

Called the Expectation Maximization Algorithm