variational inference in anglican

what is it? can we do it?

variational inference: basic idea

suppose we have a probabilistic model: $\mathbf{y}$ observed data, $\mathbf{x}$ latent random variables.

- given: joint distribution $p(\mathbf{x}, \mathbf{y})$, generally as a likelihood and prior
$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$
- desired: posterior distribution $p(\mathbf{x}|\mathbf{y})$, presumably intractable
- idea: suppose some particular parametric form for $p(\mathbf{x}|\mathbf{y})$ — call it $q(\mathbf{x}|\lambda)$ — then choose "optimal" values of parameters λ

goal: $q(\mathbf{x}|\lambda) \approx p(\mathbf{x}|\mathbf{y})$

evidence lower bound

Basic identity: decompose log marginal likelihood

$$\begin{aligned}\log p(\mathbf{y}) &= \mathcal{L}(\lambda) + D_{KL}(q||p) \\ &= \int q(\mathbf{x}|\lambda) \log \left[\frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{x}|\lambda)}\right] \mathrm{d}\mathbf{x} + \int q(\mathbf{x}|\lambda) \log \left[\frac{q(\mathbf{x}|\lambda)}{p(\mathbf{x}|\mathbf{y})}\right] \mathrm{d}\mathbf{x}\end{aligned}$$

- maximize $\mathcal{L}(\lambda)$, a lower bound on $\log p(\mathbf{y})$, w.r.t. λ
- closed-form updates if $p(\mathbf{x}, \mathbf{y})$ conjugate-exponential [beal & ghahramani 2003]
- otherwise, ???

stochastic gradient

maybe we can do stochastic gradient optimization?
[ranganath et al., 2014]

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_q \left[\log \left[\frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x}|\lambda)} \right] \nabla_\lambda \log q(\mathbf{x}|\lambda) \right]$$

- need to be able to evaluate $q(\mathbf{x}|\lambda)$ and $\nabla_\lambda \log q(\mathbf{x}|\lambda)$
- need to be able to approximate expectations over q, e.g. by sampling
- using monte carlo estimate of gradient generally thought to be basically hopeless!!

how general is this procedure?

with $\mathbf{x}^s \sim q(\mathbf{x}|\lambda)$, we have

$$\nabla_\lambda \mathcal{L}(\lambda) \approx \frac{1}{S} \sum_{s=1}^{S} \log \left[\frac{p(\mathbf{x}^s, \mathbf{y})}{q(\mathbf{x}^s|\lambda)} \right] \nabla_\lambda \log q(\mathbf{x}^s|\lambda).$$

- this works for any $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$ we can evaluate pointwise
- this works for any differentiable q
- we haven't made any exponential family assumption
- we haven't made any mean-field assumption (yet)

in particular, there is no restriction on how $p(\mathbf{x}, \mathbf{y})$ is parameterized

algorithm

basic algo block [ranganath et al., 2014]:

Algorithm 1 Black Box Variational Inference

Input: data x, joint distribution p, mean field variational family q.
Initialize $\lambda_{1:n}$ randomly, $t = 1$.
repeat
  // Draw S samples from q
  for $s = 1$ **to** S **do**
    $z[s] \sim q$
  end for
  $\rho = t$th value of a Robbins Monro sequence (Eq. 2)

  $\lambda = \lambda + \rho \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q(z[s]|\lambda)(\log p(x, z[s]) - \log q(z[s]|\lambda))$
  $t = t + 1$
until change of λ is less than 0.01.

variance reduction: control variates

control variates [ross, 2002]: we want $\mathbb{E}[f]$. consider $\hat{f}$ with

$$\hat{f}(\mathbf{x}) = f(\mathbf{x}) + a(h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])$$

if $\mathbb{E}[h(\mathbf{x})] = 0$, then

$$\mathbb{E}[\hat{f}(\mathbf{x})] = \mathbb{E}[f(\mathbf{x}) + ah(\mathbf{x})] = \mathbb{E}[f(\mathbf{x})]$$

an appropriate control variate: $h(\mathbf{x}) = \nabla_\lambda \log q(\mathbf{x})$

$$\mathbb{E}_q[\nabla_\lambda \log q(\mathbf{x}|\lambda)] = \mathbb{E}_q\left[\frac{1}{q(\mathbf{x}|\lambda)}\nabla_\lambda q(\mathbf{x}|\lambda)\right] = \nabla_\lambda \int q(\mathbf{x}|\lambda)\mathrm{d}\mathbf{x}$$

variance reduction: rao-blackwellization

rao-blackwellization: use markov blankets

- not necessary to use samples from all of $q(\mathbf{x})$
- for each x_i, if q is mean field, we may only need to sample a few other x_j to estimate that particular component of $\nabla_{\lambda_i} \mathcal{L}(\lambda)$

according to plots in the paper, much of the variance reduction is here...! [ranganath et al., 2014]

step size

stochastic optimization:

$$\lambda \leftarrow \lambda^{prev} + \rho \hat{\nabla}_\lambda \mathcal{L}(\lambda)$$

- what is ρ? what if λ is high dimensional?
- adagrad: estimate per-dimension scaling for each λ_i, track sum of squared gradient
- rmsprop: similar, but with discounting

in probabilistic programs

- goal is to get a distribution over program output: the quantity of interest may be some function $\phi(\mathbf{x})$
- deterministic computation: just define each x_i with $p(x_i|\psi_i(x_{1:i-1}))$ a la [wingate & weber 2013]
- randomness in number of parameters: might be a problem

in probabilistic programs

define the probability of a program trace as

$$p(\mathcal{P}) \propto p(\mathbf{x}, \mathbf{y}) \triangleq \prod_{i \in \mathcal{O}_x} f_i(x_i | x_{<i}) \prod_{j \in \mathcal{O}_y} g_j(y_j | x_{<j})$$

- we sample $\mathbf{x}$: each x_i is sampled from $f_i(x_i|\cdot)$
- we observe $\mathbf{y}$: each y_j has a likelihood evaluation $g_j(y_j|\cdot)$

suppose each conditional $f_i(x_i|\cdot)$ is in some known parametric family of distributions. we instead sample from a $q_i(x_i|\lambda_i)$, and compute the gradient.

mean field probabilistic program

- each time we encounter a sample checkpoint, instead of sampling x_i from the program itself we sample from some variational density $q(x_i|\lambda_i)$.
- compute proposal probability $\log q(\mathbf{x}|\lambda)$
- compute proposal gradient $\nabla_\lambda \log q(\mathbf{x}|\lambda)$
- compute trace probability $\log p(\mathbf{x}, \mathbf{y})$

this defines an approximating program which *has no observe statements*!

posterior is defined by both $q(\mathbf{x}|\lambda)$, and samples from the approximating program

in anglican

- need to implement gradients $\nabla_\lambda q(\mathbf{x}|\lambda)$ for each ERP
- ... and, actually, that's it!

minor details: sometimes take gradients w.r.t. transformed version of parameters, e.g. log standard deviation instead of standard deviation for gaussians

simple program

here's a simple model, with no deterministic computation
(will show as anglican code):

$$a \sim \mathrm{Normal}(2, 2) \qquad b \sim \mathrm{Normal}(a, 3) \qquad c \sim \mathrm{Normal}(b, 1)$$

we observe $c = 0$, and want to predict a and b.
joint density of program trace:

$$p(a, b, c) = p(c|b)p(b|a)p(a)$$

parameterization

funny "new" issue: how to parameterize latent space, if we can have any arbitrary deterministic computation?

$$a \sim \mathrm{Normal}(2, 2) \qquad b \sim \mathrm{Normal}(a, 3) \qquad c \sim \mathrm{Normal}(b, 1)$$

here's the same model, written differently:

$$\begin{aligned} a' &\sim \mathrm{Normal}(0, 1) & a &= 2a' + 2 \\ b' &\sim \mathrm{Normal}(0, 1) & b &= 2a' + 3b' + 2 \\ c &\sim \mathrm{Normal}(b, 1) \end{aligned}$$

learn the marginal, or learn the errors?
joint density of program trace:

$$p(a', b', c) = p(c|\psi(a', b'))p(b')p(a')$$

predicting output of computations

basic approach: sample from the variational program, compute $\psi(\cdot)$ of samples *does not appear to be reliable*.

what are our options?

can we only predict values which we also sample?

differentiable languages?

in this model, in anglican, we take the gradient w.r.t. a' and b':

$$
\begin{aligned}
a' &\sim \mathrm{Normal}(0,1) & \qquad a &= 2a' + 2 \\
b' &\sim \mathrm{Normal}(0,1) & \qquad b &= 2a' + 3b' + 2 \\
c &\sim \mathrm{Normal}(b,1)
\end{aligned}
$$

what if we could take the gradient w.r.t a and b? this is then like the "reparameterization trick" [kingma & welling 2014]

this seems to require, though, that all deterministic computation (i.e. our program) be differentiable. how does picture do this? [kulkarni et al. 2015]
how does stan do this? [kucukelbir et al. 2015]

efficiency?

what kind of knowledge would we need about dependency structure to use the rao-blackwellization?

- do we need full dependency graph?
- what even is the markov blanket of structural random choices?

i have no idea how to do this.