flows

HW 4: tonight

HW 5 posted, due Fri before last class ~Dec 6

COMPSCI 688: Probabilistic Graphical Models

Lecture 21: Learning with Stochastic Variational Inference

Dan Sheldon

Manning College of Information and Computer Sciences
University of Massachusetts Amherst

Partially based on materials by Benjamin M. Marlin (marlin@cs.umass.edu) and Justin Domke (domke@cs.umass.edu)

Aside: how to maximize ELBO?

stochastic gradient ascent

get unbiased estimate $\hat{\nabla} \approx \nabla$ ELBO, move in that direction

Black-Box Stochastic Variational Inference

→ reparametrization trick

Learning

$\log p_\theta(x) \geq \mathrm{ELBO}(\phi, \theta)$

max wrt ϕ: get $q_\phi(z) \approx p(z|x)$

Review: Variational Auto-Encoder

z latent code $\in \mathbb{R}^{10}$ slanty vs. straight

x image $\in \{0,1\}^{400}$

Factor analysis model with non-linear mapping

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$$

$$p(x_j|\mathbf{z}) = \text{Bernoulli}(x_j; (f_\theta(\mathbf{z}))_j), \qquad j = 1, \ldots, d$$

$\mu = f_\theta(z)$ $x_j \sim \text{Bernoulli}(\mu_j)$

Example non-linear mapping:

$$f_\theta(\mathbf{z}) = h_2(\mathbf{b}_2 + \mathbf{W}_2 \cdot h_1(\mathbf{b}_1 + \mathbf{W}_1\mathbf{z}))$$

400, 400×50, 50, 50, 50×10, 10; h_2: sigmoid; $\mathbb{R}^{10} \to \mathbb{R}^{400}$

Exact inference and learning are *intractable*.

$\theta = (b_2, W_2, b_1, W_1)$

Stochastic Variational Inference

z code, x image, p fixed; want $p(z|x) \approx q_\phi(z)$

Choose variational family, e.g., diagonal Gaussian, to approximate posterior:

$$q_\phi(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)), \quad \phi = (\boldsymbol{\mu}, \boldsymbol{\sigma})$$

$= \prod_j \mathcal{N}(\mu_j, \sigma_j^2)$ 10 10 $\sigma^2 = [\sigma_1^2, \ldots, \sigma_{10}^2]$

Stochastic optimization: repeatedly get **unbiased gradient estimate** $\hat{\nabla}_\phi$, update ϕ:

$$\hat{\nabla}_\phi \approx \nabla_\phi \text{ELBO}(\phi)$$

$$\phi \leftarrow \phi + \alpha \hat{\nabla}_\phi$$

How to get $\hat{\nabla}_\phi$?

Warmup: Estimating the ELBO

$$\mathrm{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log \frac{p(z,x)}{q_\phi(z)}\right]$$

It's easy to estimate the ELBO via Monte Carlo samples:

eval $p(z^{(i)}, x)$

$$\mathrm{ELBO}(\phi) \approx \frac{1}{K}\sum_{i=1}^{K} \log \frac{p(z^{(i)}, x)}{q_\phi(z^{(i)})} \qquad z^{(i)} \sim q_\phi \quad \leftarrow \text{sample from } q$$

eval $q_\phi(z^{(i)})$

This is unbiased: expected value of RHS = $\mathrm{ELBO}(\phi)$ for any value of K.

Estimating the Gradient? (False Start)

What happens if we try to estimate the gradient the same way? We want:

$$\nabla_\phi \mathrm{ELBO}(\phi) = \nabla_\phi \mathbb{E}_{q_\phi}\left[\log \frac{p(Z,x)}{q_\phi(Z)}\right]$$

Consider the estimate: $(K=1)$

$$\nabla_\phi \log \frac{p(z,x)}{q_\phi(z)}, \quad z \sim q_\phi.$$

$$\approx$$

$$\mathbb{E}_{q_\phi(z)}\left[\nabla_\phi \log \frac{p(z,x)}{q_\phi(z)}\right]$$

This is not unbiased! It neglects the fact that the *distribution* of z depends on ϕ:

$$\nabla_\phi \mathrm{ELBO}(\phi) = \nabla_\phi \int q_\phi(z) \log \frac{p(z,x)}{q_\phi(z)} dz$$

The false start incorrectly interchanges the gradient and the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi}\left[\log \frac{p(Z,x)}{q_\phi(Z)}\right] \neq \mathbb{E}_{q_\phi}\left[\nabla_\phi \log \frac{p(Z,x)}{q_\phi(Z)}\right]$$

$$= \int \nabla_\phi \left(q_\phi(z) \log \frac{p(z,x)}{q_\phi(z)}\right) dz \rightarrow \text{product rule}$$

"score function estimator"

Reparameterization Trick $\quad \mathbb{E}_{q_\phi}[\dots] \xrightarrow{\text{trick}} \mathbb{E}_{q(\epsilon)}[\dots]$

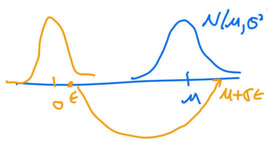
fixed

The reparameterization trick is a way to convert the ELBO into an expectation with respect to a *fixed* distribution (independent of ϕ) so we can interchange the gradient and expectation. The idea is to draw samples of z by transforming a random variable from a fixed base distribution.

"noise" $\quad q_\phi$

Example: $z = \mu + \sigma\epsilon, \ \epsilon \sim \mathcal{N}(0,1) \implies z \sim \mathcal{N}(\mu, \sigma^2)$

$q(\epsilon) \qquad q_\phi(z)$



General case: $z = \mathcal{T}_\phi(\epsilon), \ \epsilon \sim q(\epsilon) \implies z \sim q_\phi(z)$

We call $\mathcal{T}_\phi$ and $q(\epsilon)$ a *reparameterization* of q_ϕ

works: location-scale $\qquad$ normalizing
(easily) $\quad$ elliptical $\qquad$ flows

ELBO Gradient with Reparameterization

$\mathbb{E}_{q_\phi(z)}\left[\log \frac{p(z,x)}{q_\phi(z)}\right]$

With reparameterization, we can write the ELBO as an expectation over $q(\epsilon)$:

$z = T_\phi(\epsilon)$, $\epsilon \sim q(\epsilon)$

$$\text{ELBO}(\phi) = \mathbb{E}_{q(\epsilon)}\left[\log \frac{p(\overbrace{\mathcal{T}_\phi(\epsilon)}^{z}, x)}{q_\phi(\underbrace{\mathcal{T}_\phi(\epsilon)}_{z})}\right]$$

Now we can interchange the gradient and expectation

$$\nabla_\phi \text{ELBO}(\phi) = \nabla_\phi \mathbb{E}_{q(\epsilon)}\left[\log \frac{p(\mathcal{T}_\phi(\epsilon), x)}{q_\phi(\mathcal{T}_\phi(\epsilon))}\right] = \mathbb{E}_{q(\epsilon)}\left[\nabla_\phi \log \frac{p(\mathcal{T}_\phi(\epsilon), x)}{q_\phi(\mathcal{T}_\phi(\epsilon))}\right]$$

$$\nabla_\phi \int q(\epsilon) \log \frac{p(T_\phi(\epsilon),x)}{q_\phi(T_\phi(\epsilon))}\, d\epsilon = \int q(\epsilon)\, \nabla_\phi \log \frac{p(T_\phi(\epsilon),x)}{q_\phi(T_\phi(\epsilon))}\, d\epsilon$$

Reparameterization Gradient Estimate

$\log q_\phi(z) = \log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(z-\mu)^2$

$\log q_\phi(\mu+\sigma\epsilon) = \log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(\mu+\sigma\epsilon-\mu)^2$

This gives a simple unbiased Monte Carlo estimate of the gradient:

$$g = \nabla_\phi \underbrace{\left(\frac{1}{K}\sum_{i=1}^{K} \log \frac{p\left(\mathcal{T}_\phi(\epsilon^{(i)}), x\right)}{q_\phi\left(\mathcal{T}_\phi(\epsilon^{(i)})\right)}\right)}_{\text{ELBO}(\phi, \epsilon^{(1)}, \ldots, \epsilon^{(K)})}, \qquad \epsilon^{(1)}, \ldots, \epsilon^{(K)} \sim q(\epsilon)$$

We can compute it as follows:

1. Draw $\epsilon^{(1)}, \ldots, \epsilon^{(K)} \sim q(\epsilon)$
2. Compute $\widehat{\text{ELBO}}(\phi, \epsilon^{(1)}, \ldots, \epsilon^{(K)})$ = term in parentheses above
3. Use autodiff to get $g = \nabla_\phi \widehat{\text{ELBO}}(\phi, \epsilon^{(1:K)})$

Pytorch, JAX

Gradient Estimation: Reparameterization

	Without reparameterization	With reparameterization
Variational distribution	$q_\phi(z)$	$q_\phi(z)$
Sampling	$z \sim q_\phi(z)$	$\epsilon \sim q(\epsilon), z = \mathcal{T}_\phi(\epsilon)$
ELBO	$\mathbb{E}_{q_\phi(Z)}\left[\log \frac{p(Z,x)}{q_\phi(Z)}\right]$	$\mathbb{E}_{q(\epsilon)}\left[\log \frac{p(\mathcal{T}_\phi(\epsilon),x)}{q_\phi(\mathcal{T}_\phi(\epsilon))}\right]$
ELBO estimate	$\log \frac{p(z,x)}{q_\phi(z)}, z \sim q_\phi(z)$	$\log \frac{p(\mathcal{T}_\phi(\epsilon),x)}{q_\phi(\mathcal{T}_\phi(\epsilon))}, \epsilon \sim q(\epsilon)$
Gradient estimate	$\nabla_\phi \log \frac{p(z,x)}{q_\phi(z)}, z \sim q_\phi(z)$ (wrong/biased)	$\nabla_\phi \log \frac{p(\mathcal{T}_\phi(\epsilon),x)}{q_\phi(\mathcal{T}_\phi(\epsilon))}, \epsilon \sim q(\epsilon)$ (unbiased)

Reparameterization with Diagonal Gaussians

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_d^2 \end{bmatrix}$$

Suppose the variational family is a diagonal Gaussian

$$q_\phi(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$$

This can be reparameterized as:

$z_j = \mu_j + \sigma_j \epsilon_j \qquad \epsilon_j \sim N(0,1)$

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon, \quad \underbrace{\epsilon \sim \mathcal{N}(0, I)}$$

$= T_\phi(\epsilon)$

($\odot$ = elementwise multiplication)

Aside: Reparameterization with Arbitrary Gaussians

Another choice would be to use a general Gaussian distribution:

$$\epsilon \sim \mathcal{N}(0, I) \implies \mu + L\epsilon \sim \mathcal{N}(\mu, LL^\top).$$

This is a reparameterization with

$$q(\epsilon) = \mathcal{N}(\epsilon|0, I), \qquad \mathcal{T}_\phi(\epsilon) = \mu + L\epsilon \qquad \phi = (L, \mu)$$

It covers any multivariate Gaussian, since an arbitrary covariance matrix Σ can be written as $\Sigma = LL^\top$ for some L (e.g., a Cholesky factor)

Example: Bernoulli VAE

θ

$W = [w_1 \dots w]\, z$

$z \to x$ $\{0,1\}^{400}$

Let's return to our Bernoulli VAE factor analysis model and use a diagonal Gaussian approximation:

model p (fixed)

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$$
$$p(x_j|\mathbf{z}) = \text{Bernoulli}(x_j; (f_\theta(\mathbf{z}))_j), \qquad j = 1, \dots, d$$

variational

$$q_\phi(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$$

BBSVI would repeat the following steps:

Params $\phi = (\overset{10}{\mu}, \overset{10}{\sigma^2})$

Init $\phi = (\mu, \sigma)$

$\epsilon \sim \mathcal{N}(0, I)$ $\qquad \log \frac{p(z,x)}{q_\phi(z)}$ $\qquad z = \mathcal{T}_\phi(\epsilon)$ $\quad (\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon)$

$$\hat{\nabla}_{\mu,\sigma} = \nabla_{\mu,\sigma} \Big\{ \log \mathcal{N}(\overset{z}{\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon};\ 0, I) \qquad \log p(z)$$

$$+ \sum_{j=1}^{d} \log \text{Bernoulli}\Big(x_j; (f_\theta(\overset{z}{\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon}))_j\Big) \qquad + \log p(x|z)$$

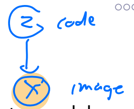
$$- \log \mathcal{N}\Big(\overset{z}{\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon};\ \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)\Big) \Big\} \qquad - \log q_\phi(z)$$

$$(\boldsymbol{\mu}, \boldsymbol{\sigma}) \leftarrow (\boldsymbol{\mu}, \boldsymbol{\sigma}) + \alpha \cdot \hat{\nabla}_{\mu,\sigma}$$

With the optimized parameters we could approximate $p(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z})$ and lower bound the log-marginal likelihood $\log p(\mathbf{x}) \geq \text{ELBO}(\phi)$. **What about learning?**

Learning with Stochastic Variational Inference

Learning with Stochastic Variational Inference



The basic idea is to jointly maximize the ELBO with respect to model parameters θ and variational parameters ϕ by getting unbiased gradient estimates for both:

$$\log p_\theta(\mathbf{x}) \geq \text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(Z, x)}{q_\phi(Z)}\right]$$

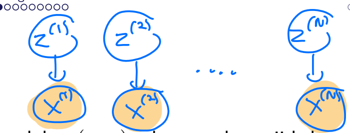
$$\hat{\nabla}_\theta \approx \nabla_\theta \text{ELBO}(\theta, \phi)$$

$$\hat{\nabla}_\phi \approx \nabla_\phi \text{ELBO}(\theta, \phi)$$

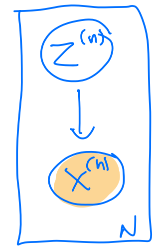
$$(\theta, \phi) \leftarrow (\theta, \phi) + \alpha \cdot (\hat{\nabla}_\theta, \hat{\nabla}_\phi)$$

Learning with IID Data

$\max \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x^{(n)})$



How do we learn a latent variable model $p_\theta(\mathbf{z}, \mathbf{x})$ when we have iid data $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$?



Each datum $\mathbf{x}^{(n)}$ has its own:

- marginal likelihood $p_\theta(\mathbf{x}^{(n)})$
  $p_\theta(x^{(n)} \mid z^{(n)})$
- posterior $p_\theta(\mathbf{z}^{(n)} \mid \mathbf{x}^{(n)})$
- **variational distribution** $q_{\phi^{(n)}}(\mathbf{z}^{(n)})$

Learning with IID Data

Images: $x^{(1)}, \ldots, x^{(N)}$

Basic approach: introduce variational parameters $\phi^{(n)}$ for each datum and construct an overall lower bound:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\log p_\theta(\mathbf{x}^{(n)})} \geq \frac{1}{N} \sum_{n=1}^{N} \underbrace{\text{ELBO}(\theta, \phi^{(n)}, \mathbf{x}^{(n)})}$$

$$\text{ELBO}(\theta, \phi^{(n)}, \mathbf{x}^{(n)}) = \mathbb{E}_{q_{\phi^{(n)}}}\left[\log p_\theta(\mathbf{Z}^{(n)}, \mathbf{x}^{(n)}) - \log q_{\phi^{(n)}}(\mathbf{Z}^{(n)})\right]$$

$q_{\phi^{(n)}}(z^{(n)}) \approx p_\theta(z^{(n)} \mid x^{(n)}) \quad \forall n$

Find θ to assign high prob to $x^{(1)}, \ldots, x^{(N)}$

Then optimize the lower bound with respect to all parameters. Compute:

$$\hat{\nabla}_{\phi^{(n)}} \approx \nabla_{\phi^{(n)}} \text{ELBO}(\theta, \phi^{(n)}, \mathbf{x}^{(n)}), \quad n = 1, \ldots, N,$$

$$\hat{\nabla}_\theta \approx \nabla_\theta \frac{1}{N} \sum_{n=1}^{N} \text{ELBO}(\theta, \phi^{(n)}, \mathbf{x}^{(n)})$$

Then update $\theta, \underbrace{\phi^{(1)}, \ldots, \phi^{(N)}}$ using stochastic gradients.

gen model

variational params, for each image

Amortized Inference

The basic approach described above introduces a very large number of variational parameters and can be very slow for large data sets.

Amortized inference proposes to use a neural net to *predict* the variational parameters $\phi^{(n)}$ for datum $\mathbf{x}^{(n)}$, e.g.

$$q_{\phi^{(n)}}(z^{(n)}) = \mathcal{N}(z^{(n)};\ \mu^{(n)}, \tau^2 I)$$

old: separate learned mean per image

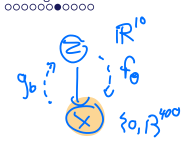
$\phi^{(n)} = \mu^{(n)}$

$\phi = (\mu^{(1)}, \dots, \mu^{(N)}, \tau^2)$

$$q_\phi(\mathbf{z}^{(n)} \mid \mathbf{x}^{(n)}) = \mathcal{N}(\mathbf{z}^{(n)};\ g_\phi(\mathbf{x}^{(n)}), \tau^2 I)$$

- The function g_ϕ predicts the mean of the variational posterior approximation for datum $\mathbf{x}^{(n)}$. (We could also model the (co)variance as some function of $\mathbf{x}^{(n)}$.)
- This is called *amortization* because it shares information across data points for learning the variational approximations.

Amortized Inferences: VAEs



$f_\theta : z \longmapsto \mu_x$

$g_\phi : x \longmapsto \mu_z$

A common choice for g_ϕ is a multi-layer neural network, similar to f_θ, e.g.:

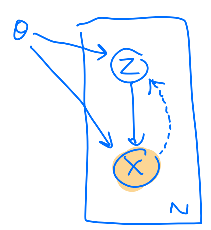
$$f_\theta(\mathbf{z}) = \overset{\text{sigmoid}}{h_2}(\mathbf{b}_2 + \mathbf{W}_2 \cdot \overset{\text{ReLu}}{h_1}(\mathbf{b}_1 + \mathbf{W}_1 \overset{\text{code}}{\mathbf{z}}))$$

$$g_\phi(\mathbf{x}) = \mathbf{b}_4 + \mathbf{W}_4 \cdot \overset{\text{ReLu}}{h_3}(\mathbf{b}_3 + \mathbf{W}_3 \underset{\text{image}}{\mathbf{x}})$$

ϕ

(h_1, h_2, h_3 are elementwise non-linear functions)

Illustration: p and q graphical models

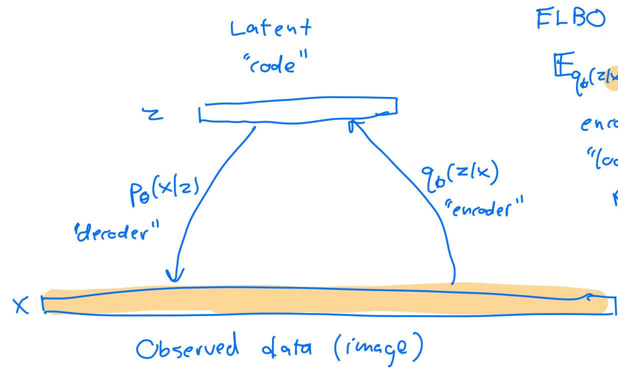


p_θ

——— p_θ

- - - - - - $q_\phi(z|x)$ (no joint dist, only the conditional $q_\phi(z|x)$)

Illustration: "Auto-Encoder"



ELBO "energy-term"

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

encourages encoding/decoding "loop" to approximately preserve the image

Example: Inference and Learning in Bernoulli VAE

Putting all the pieces together, stochastic variational inference and learning for a Bernoulli VAE would repeat the following for all n in some order:

$\epsilon \sim \mathcal{N}(0, I)$ $\quad q_\phi(z^{(n)} | x^{(n)}) = \mathcal{N}(z^{(n)}; g_\phi(x^{(n)}), \tau^2 I)$ $\quad (\mathbf{z}^{(n)} = g_\phi(\mathbf{x}^{(n)}) + \tau\epsilon)$

$$
\begin{aligned}
\hat{\nabla}_{\theta,\phi} = \nabla_{\theta,\phi} \Big\{ & \log \mathcal{N}\left(g_\phi(\mathbf{x}^{(n)}) + \tau\epsilon;\ 0, I\right) && \log p(z^{(n)}) \\
& + \sum_{j=1}^{d} \log \text{Bernoulli}\left(x_j^{(n)};\ \left(f_\theta(g_\phi(\mathbf{x}^{(n)}) + \tau\epsilon)\right)_j\right) && + \log p(x^{(n)} | z^{(n)}) \\
& - \log \mathcal{N}\left(g_\phi(\mathbf{x}^{(n)}) + \tau\epsilon;\ g_\phi(\mathbf{x}^{(n)}), \tau^2 I\right) \Big\} && - \log q_\phi(z^{(n)} | x^{(n)})
\end{aligned}
$$

MC approx of ELBO

$$(\theta, \phi) \leftarrow (\theta, \phi) + \alpha \cdot \hat{\nabla}_{\theta,\phi}$$

Bonus: Closed Form Entropy, Etc.

Bonus: Handling Some Terms in Closed Form

The ELBO can be decomposed into several terms with different computational properties:

$$
\begin{aligned}
\text{ELBO}(\phi) &= \mathbb{E}_{q_\phi}\left[\log \frac{p(Z, x)}{q_\phi(Z)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi}[\log p(Z)]}_{\text{"cross entropy"}} + \underbrace{\mathbb{E}_{q_\phi}[\log p(x|Z)]}_{\text{"energy"}} - \underbrace{\mathbb{E}_{q_\phi}[\log q_\phi(Z)]}_{\text{"entropy"}}
\end{aligned}
$$

With simple distributions (esp. Gaussians) the cross entropy and entropy terms can often be computed in closed form.

Example: Closed-Form Cross-Entropy

Example: $p(\mathbf{z})$ is a standard normal and q_ϕ is a diagonal Gaussian:

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, I) \\ q_\phi(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)) \end{aligned} \implies \int q_\phi(\mathbf{z}) \log p(\mathbf{z}) \, dz = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{d}(\mu_j^2 + \sigma_j^2)$$

$z_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$

When possible, it's usually (but not always) best to compute these terms and their gradients analytically, and only use Monte Carlo estimation for the energy term.

This is because lower variance gradient estimates will make the stochastic optimization converge faster.