Biostatistics 682: Applied Bayesian Inference Lecture 12: Variational Inference: A Fast Alternative to MCMC

Jian Kang

Department of Biostatistics
University of Michigan, Ann Arbor

Outline

- Explain the optimization view of Bayesian inference using $\mathrm{KL}\big(q \,\|\, \pi(\cdot \mid y)\big)$.
- Derive the Evidence Lower BOund (ELBO) using $\pi(\cdot)$ notation.
- Implement mean-field Coordinate Ascent Variational Inference (CAVI) for Bayesian linear regression.
- Contrast Variational Inference (VI) with Markov Chain Monte Carlo (MCMC) in terms of accuracy and speed. Inference (VI).

Motivation: When MCMC Struggles

Context: MCMC is asymptotically exact but may be too slow for large n, high p, or complex posteriors.

Idea of VI: Replace sampling with optimization: choose a family $\mathcal{Q}$ and find $q^* \in \mathcal{Q}$ closest to the posterior.

$$q^* \; = \; \arg\min_{q \in \mathcal{Q}} \; \mathrm{KL}\big(q(\theta) \,\|\, \pi(\theta \mid y)\big).$$

From Marginal Likelihood to ELBO

Posterior and evidence (density notation π):

$$\pi(\theta \mid y) \;=\; \frac{\pi(y,\theta)}{\pi(y)}, \qquad \pi(y) \;=\; \int \pi(y,\theta)\,d\theta.$$

Key identity:

$$\begin{aligned} \log \pi(y) &= \mathcal{L}(q) + \mathrm{KL}\big(q(\theta) \,\|\, \pi(\theta \mid y)\big), \\ \mathcal{L}(q) &= \mathbb{E}_q\big[\log \pi(y,\theta)\big] - \mathbb{E}_q\big[\log q(\theta)\big]. \end{aligned}$$

Maximizing $\mathcal{L}(q)$ is equivalent to minimizing $\mathrm{KL}(q \,\|\, \pi(\cdot \mid y))$.

What is Kullback–Leibler (KL) Divergence?

Definition:

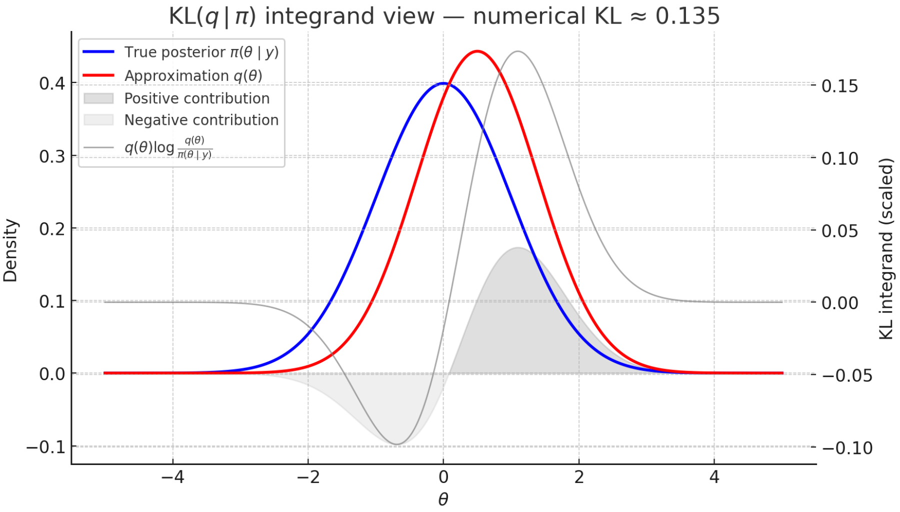
$$\mathrm{KL}\big(q(\theta) \,\|\, \pi(\theta \mid y)\big) = \int q(\theta) \log \frac{q(\theta)}{\pi(\theta \mid y)} \, d\theta.$$

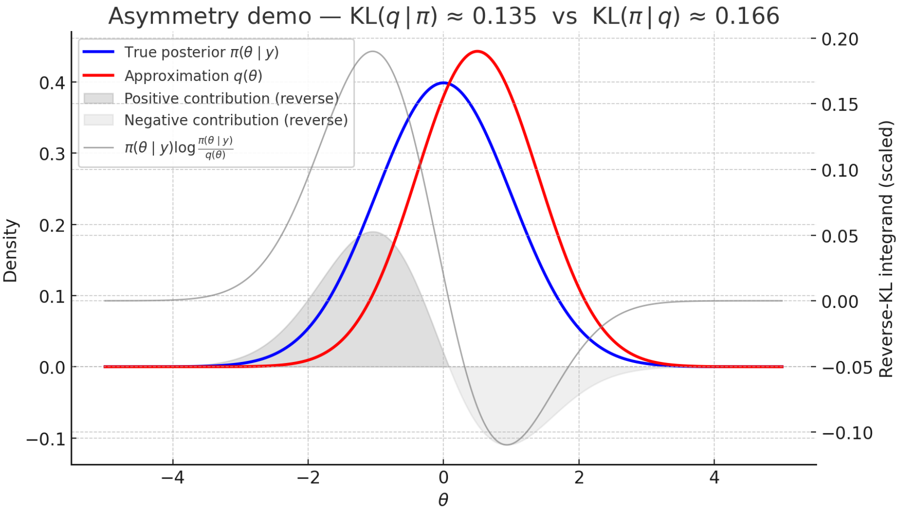
Intuition:

- Measures how different the approximate distribution $q(\theta)$ is from the true posterior $\pi(\theta \mid y)$.
- Always nonnegative: $\mathrm{KL}(q \,\|\, \pi) \geq 0$, and equals 0 only if $q = \pi$.
- Asymmetric: $\mathrm{KL}(q \,\|\, \pi) \neq \mathrm{KL}(\pi \,\|\, q)$.

Interpretation:

- In Variational Inference, we minimize $\mathrm{KL}(q \,\|\, \pi)$ to make $q(\theta)$ close to $\pi(\theta \mid y)$.
- Think of it as the "information loss" when using q to approximate π.


KL($q\,|\,\pi$) integrand view — numerical KL ≈ 0.135
True posterior $\pi(\theta \mid y)$
Approximation $q(\theta)$
Positive contribution
Negative contribution
$q(\theta)\log\frac{q(\theta)}{\pi(\theta \mid y)}$
Density
KL integrand (scaled)
θ
0.4
0.3
0.2
0.1
0.0
−0.1
0.15
0.10
0.05
0.00
−0.05
−0.10
−4
−2
0
2
4


Asymmetry demo — KL($q\,|\,\pi$) ≈ 0.135 vs KL($\pi\,|\,q$) ≈ 0.166
True posterior $\pi(\theta \mid y)$
Approximation $q(\theta)$
Positive contribution (reverse)
Negative contribution (reverse)
$\pi(\theta \mid y)\log\frac{\pi(\theta \mid y)}{q(\theta)}$
Density
Reverse-KL integrand (scaled)
θ
0.4
0.3
0.2
0.1
0.0
−0.1
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−4
−2
0
2
4

Mean-Field Approximation and CAVI

Mean-field factorization:

$$q(\theta) = \prod_{j=1}^{J} q_j(\theta_j).$$

Coordinate ascent updates:

$$\log q_j^*(\theta_j) \;=\; \mathbb{E}_{q_{-j}}\left[\log \pi(y,\theta)\right] + \text{const.}$$

Notes:

- For conjugate-exponential families, q_j is in the same family as its full conditional.
- Each step increases the ELBO; iterate to convergence.

Example Model: Bayesian Linear Regression

Model (with densities π):

$$
\begin{aligned}
&y \in \mathbb{R}^n,\ \ X \in \mathbb{R}^{n\times p},\\
&\pi(y \mid X, \beta, \sigma^2) = \mathcal{N}\big(y\,;\, X\beta,\, \sigma^2 I\big),\\
&\pi(\beta) = \mathcal{N}\big(\beta\,;\, 0,\, \tau^2 I\big), \qquad \pi(\sigma^{-2}) = \mathrm{G}(a, b).
\end{aligned}
$$

Variational family: $q(\beta, \sigma^{-2}) = q(\beta)\, q(\sigma^{-2})$.

Optimal $q(\beta)$:

$$\log q^*(\beta) = \mathbb{E}_{q(\sigma^{-2})}\left[\log \pi(y, \beta, \sigma^2)\right] + \text{const.}$$

Result: $q(\beta) = \mathcal{N}(m, \Sigma)$ with

$$\Sigma = \left(\mathbb{E}_{q(\sigma^{-2})}\left[\sigma^{-2}\right] X^\top X + \tau^{-2} I\right)^{-1},$$
$$m = \Sigma\, \mathbb{E}_{q(\sigma^{-2})}\left[\sigma^{-2}\right] X^\top y.$$

If $q(\sigma^{-2}) = \mathrm{G}(a', b')$, then $\mathbb{E}[\sigma^{-2}] = a'/b'$.

CAVI Derivations for Linear Regression (II)

Optimal $q(\sigma^{-2})$:

$$\log q^*(\sigma^{-2}) = \mathbb{E}_{q(\beta)}\big[\log \pi(y, \beta, \sigma^2)\big] + \text{const.}$$

Result: $q(\sigma^{-2}) = \mathrm{G}(a', b')$ with

$$\begin{aligned} a' &= a + \tfrac{n}{2}, \\ b' &= b + \tfrac{1}{2}\,\mathbb{E}_{q(\beta)}\big(\|y - X\beta\|^2\big), \\ \mathbb{E}_{q(\beta)}\big(\|y - X\beta\|^2\big) &= \|y - Xm\|^2 + \mathrm{tr}\big(X^\top X\,\Sigma\big). \end{aligned}$$

CAVI Algorithm (Pseudo-code)

Initialization: choose a', b', m, Σ.

Repeat until convergence:

1. Update $\Sigma \leftarrow \left((a'/b')X^\top X + \tau^{-2} I\right)^{-1}$.
2. Update $m \leftarrow \Sigma\,(a'/b')X^\top y$.
3. Compute $SSE = \|y - Xm\|^2 + \mathrm{tr}(X^\top X\,\Sigma)$.
4. Update $a' \leftarrow a + n/2$, $\;b' \leftarrow b + \frac{1}{2}SSE$.
5. Monitor ELBO $\mathcal{L}(q)$ for convergence.

Implementation (Part I): Function Definition

Mean-field CAVI for Bayesian Linear Regression

```
def cavi_blr(X, y, tau2=100.0, a=1.0, b=1.0, max_iter=1000, tol=1e-6):
    n, p = X.shape
    XtX = X.T @ X
    Xty = X.T @ y

    # Initialize variational parameters
    a_p = a + n / 2.0
    m = np.zeros(p)
    Sigma = np.eye(p)
    b_p = b + 0.5 * (y @ y)  # crude init

    for _ in range(max_iter):
        inv_sigma2_mean = a_p / b_p
        # q(beta): Normal(m, Sigma)
        Sigma_new = np.linalg.inv(inv_sigma2_mean * XtX + (1.0 / tau2) *
 np.eye(p))
        m_new = Sigma_new @ (inv_sigma2_mean * Xty)
```

Notes:

- Updates for $q(\beta)$ and $q(\sigma^{-2})$ alternate until convergence.
- m and Σ are the mean and covariance of $q(\beta)$.
- a_p, b_p are shape and scale parameters of $q(\sigma^{-2}) = \mathrm{G}(a_p, b_p)$.

Implementation (Part II): Complete Function

CAVI updates for $q(\sigma^{-2})$ and convergence check:

```
        # E[||y - X beta||^2] under q(beta)
        sse = np.sum((y - X @ m_new) ** 2) + np.trace(XtX @ Sigma_new)

        # q(sigma^{-2}): G(a', b')
        b_p_new = b + 0.5 * sse

        if np.max(np.abs(m_new - m)) < tol and abs(b_p_new - b_p) < tol:
            m, Sigma, b_p = m_new, Sigma_new, b_p_new
            break

        m, Sigma, b_p = m_new, Sigma_new, b_p_new

    return m, Sigma, a_p, b_p
```

Notes:

- The function returns variational posteriors $q(\beta)$ and $q(\sigma^2)$.
- Convergence criterion checks small parameter changes.

Implementation (Part III): Running the Code

Example: run the algorithm and print results

```
import time

start = time.time()
m, Sigma, a_p, b_p = cavi_blr(
    X, y, tau2=tau**2, a=1.0, b=1.0, max_iter=2000
)
run_time = time.time() - start
print(f"Runtime: {run_time:.3f}s, a'={a_p:.3f}, b'={b_p:.3f}")
```

Outputs:

- m: posterior mean of $q(\beta)$; Σ: posterior covariance.
- a_p, b_p: parameters of $q(\sigma^{-2})$.
- Runtime summary for performance comparison with MCMC and ADVI.

ELBO (Part I)

Model with precision:

$$\pi(y \mid \beta, \sigma^{-2}) = \mathcal{N}\big(y;\, X\beta,\ \sigma^2 I\big), \quad \pi(\beta) = \mathcal{N}(0, \tau^2 I), \quad \pi(\sigma^{-2}) = \mathrm{G}(a, b).$$

Mean-field family: $q(\beta, \sigma^{-2}) = q(\beta)\, q(\sigma^{-2})$ with $q(\beta) = \mathcal{N}(m, \Sigma)$ and $q(\sigma^{-2}) = \mathrm{G}(a', b')$.

ELBO decomposition:

$$\mathcal{L}(q) = \underbrace{\mathbb{E}_q[\log \pi(y \mid \beta, \sigma^{-2})]}_{\text{fit}} + \underbrace{\mathbb{E}_q[\log \pi(\beta)] + \mathbb{E}_q[\log \pi(\sigma^{-2})]}_{\text{priors}}$$
$$- \underbrace{\mathbb{E}_q[\log q(\beta)] - \mathbb{E}_q[\log q(\sigma^{-2})]}_{\text{entropies}}.$$

Closed forms (I):

$$\mathbb{E}_q[\log \pi(y \mid \beta, \sigma^{-2})] = -\tfrac{n}{2} \log(2\pi) + \tfrac{n}{2}\, \mathbb{E}_q[\log \sigma^{-2}] - \tfrac{1}{2}\, \mathbb{E}_q[\sigma^{-2}]\, \mathbb{E}_q[\|y - X\beta\|^2],$$
$$\mathbb{E}_q[\log \pi(\beta)] = -\tfrac{p}{2} \log(2\pi\tau^2) - \tfrac{1}{2\tau^2}\, \mathbb{E}_q[\|\beta\|^2].$$

Substitutions:

$$\mathbb{E}_q[\|y - X\beta\|^2] = \|y - Xm\|^2 + \mathrm{tr}(X^\top X\, \Sigma), \qquad \mathbb{E}_q[\|\beta\|^2] = m^\top m + \mathrm{tr}(\Sigma).$$

ELBO (Part II)

Closed forms (II):

$$\begin{aligned}
\mathbb{E}_q[\log \pi(\sigma^{-2})] &= a \log b - \log \Gamma(a) + (a-1)\,\mathbb{E}_q[\log \sigma^{-2}] - b\,\mathbb{E}_q[\sigma^{-2}], \\
\mathbb{E}_q[\log q(\beta)] &= -\tfrac{1}{2} \log\left|2\pi e\, \Sigma\right|, \\
\mathbb{E}_q[\log q(\sigma^{-2})] &= a' \log b' - \log \Gamma(a') + (a'-1)\,\mathbb{E}_q[\log \sigma^{-2}] - b'\,\mathbb{E}_q[\sigma^{-2}].
\end{aligned}$$

Gamma expectations: $\mathbb{E}_q[\sigma^{-2}] = \dfrac{a'}{b'}, \quad \mathbb{E}_q[\log \sigma^{-2}] = \psi(a') - \log b'.$

Usage:

- Evaluate $\mathcal{L}(q)$ each iteration (CAVI or ADVI) to monitor convergence.

Automatic Differentiation Variational Inference

Motivation:

- Classical mean-field VI requires model-specific algebra for $\mathbb{E}_q[\log \pi(y, \theta)]$.
- ADVI automates this process using automatic differentiation and stochastic gradient optimization.

Core ideas:

- Express the ELBO as an expectation and estimate it with Monte Carlo samples.
- Reparameterize latent variables: $\theta = f(\phi, \epsilon)$ where $\epsilon \sim \mathcal{N}(0, I)$.
- Optimize variational parameters ϕ (mean and scale) using gradients computed by automatic differentiation.
- Works for both mean-field and full-rank Gaussian families.

Advantages:

- Model-agnostic: applies to any differentiable probabilistic model.
- Scalable: supports mini-batching and stochastic optimization.
- Implemented in PyMC, Stan, TensorFlow Probability, and Pyro.

ADVI: Algorithm

Algorithm summary:

1. Initialize variational parameters ϕ_0 (mean and log-scale of Gaussian).
2. Sample $\epsilon^{(s)} \sim \mathcal{N}(0, I)$ and compute $\theta^{(s)} = f(\phi, \epsilon^{(s)})$.
3. Estimate the stochastic gradient $\nabla_\phi \mathcal{L}(q_\phi)$ using automatic differentiation.
4. Update ϕ using Adam or another gradient-based optimizer.
5. Continue until the ELBO converges (stabilizes).

Limitations:

- Often assumes a Gaussian variational family $q(\theta)$ (mean-field or full-rank).
- Can underestimate or overestimate posterior uncertainty and get stuck in local optima.
- Convergence quality depends on ELBO estimation noise and initialization.

Implementation and comparisons in PyMC

Idea: Specify $\pi(y, \theta)$ once, then run either:

- MCMC (NUTS): asymptotically exact sampling from $\pi(\theta \mid y)$.
- Variational: optimization; ADVI .

Same model in PyMC; run both:

```
with pm.Model() as blr:
    beta  = pm.Normal("beta", mu=np.zeros(p), sigma=10.0, shape=p)
    sigma = pm.HalfNormal("sigma", sigma=1.0)
    y_like = pm.Normal("y", mu=pm.math.dot(X,beta), sigma=sigma, observed=y)

    idata_mcmc = pm.sample(draws=2000, tune=1000, chains=4, target_accept=0.9)
    approx = pm.fit(n=20000, method="advi")
    idata_vi = approx.sample(2000)
```