COMPSCI 689
Lecture 21: Markov Chain Monte Carlo for Approximate Bayesian Inference

Brendan O'Connor

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Problems with Bayesian Inference

- We have explored an example that admits analytic parameter posteriors and posterior predictive distributions.
- This will be the case when conjugate priors are used. This is convenient when it works, but not all likelihoods have conjugate priors.
- Important examples include logistic regression, deep neural network models, mixture models, and many more.
- The intractability of the integrals required to carry out Bayesian inference is the key challenge in applying Bayesian methods to these models.

Monte Carlo Methods

- Monte Carlo methods are one important class of solutions to the Bayesian computation problem for more complex models.
- The basic Monte Carlo approximation result is the following:

$$\mathbb{E}_{P(X)}[f(X)] = \int P(X = x) f(x) dx \approx \frac{1}{S} \sum_{s=1}^{S} f(x_s), \quad x_s \sim P(X)$$

- This result says that we can approximate an expectation with respect to a distribution $P(X)$ by drawing independent samples from $P(X)$ and computing an average over the samples.
- Such an approximation is unbiased for any S and the weak law of large numbers states that such an average will converge to the corresponding expectation as S increases.

Monte Carlo Methods for Prediction

- An important application of Monte Carlo approximation is approximating the posterior predictive distribution.
- Recall that the general posterior predictive distribution in the supervised case is given by:

$$P(Y = y|\mathbf{x}, \mathcal{D}) = \int P(Y = y|\mathbf{x}, \theta) P(\theta|\mathcal{D}) d\theta$$

- If the integral is intractable, but if we can draw samples θ_s from $P(\theta|\mathcal{D})$, we can approximate the posterior predictive distribution using:

$$P(Y = y|\mathbf{x}, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} P(Y = y|\mathbf{x}, \theta_s)$$

Monte Carlo Methods for Machine Learning

- Work on sampling methods for large-scale models is an important open area in machine learning.
- Classical Markov Chain Monte Carlo samplers include the Metropolis-Hastings sampler and the Gibbs sampler. More recent examples include gradient-based methods, like Hamiltonian Monte Carlo.

The Metropolis-Hastings Sampler

Algorithm 24.2: Metropolis Hastings algorithm

Initialize x^0 ;

for $s = 0, 1, 2, \ldots$ **do**

  Define $x = x^s$;

  Sample $x' \sim q(x'|x)$;

  Compute acceptance probability

$$\alpha = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$

  Compute $r = \min(1, \alpha)$;

  Sample $u \sim U(0, 1)$;

  Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if } u < r \\ x^s & \text{if } u \geq r \end{cases}$$

MCMC in a nutshell

- Detailed balance conditions imply the Markov chain's stationary state draws from the target distribution. So the sample chain $\{x^s\}_{s=1..S}$ are draws from $p(x)$.
- For a long chain, you should stochastically explore the *entire* space, not just one mode. In theory you will. ("Convergence" of the chain; not the parameter state as in optimization.)
- In practice, not always. It's possible to diagnose a chain failing to converge, but impossible to be sure it's properly converged.
- On the plus side, MCMC is a mature technology with extensive resources and heuristics to diagnose such issues (e.g. multiple chain convergence; see books like *Bayesian Data Analysis*).

The Gibbs Sampler

Algorithm 1 Gibbs sampler

Initialize $x^{(0)} \sim q(x)$

for iteration $i = 1, 2, \ldots$ **do**

$$x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$$

$$x_2^{(i)} \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$$

$$\vdots$$

$$x_D^{(i)} \sim p(X_D = x_D | X_1 = x_1^{(i)}, X_2 = x_2^{(i)}, \ldots, X_D = x_{D-1}^{(i)})$$

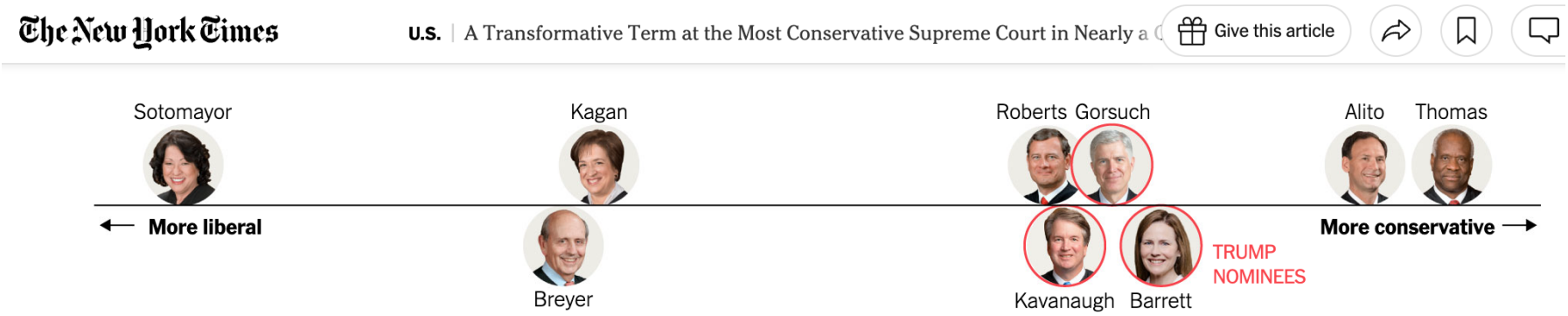
end for

- Useful when individual $p(x_i \mid x_{-i})$ posteriors are tractable, conjugate, or discrete.

Example: Gaussian Mixture

- Goal: fully Bayesian inference for both z and parameters π, μ, σ.
- Model: $z_i \sim Categ(\pi)$, $x_{i,d} \mid z_i \sim N(\mu_{z_i,d}, \sigma^2_{z_i,d})$, and diffuse conjugate priors $\mu_{k,d} \sim N(0, 100)$, $\sigma^2_{k,d} \sim IG(3, .001)$, $\pi \sim Dir(\frac{1}{K})$
- Sampler: initialize all z, μ, σ^2 then iterate:
  - For all i, $z_i \sim p(z_i \mid x_i, \mu, \sigma^2)$ *[Same as EM E-step]*
  - For each k, $\mu_k \sim p(\mu_k \mid \{x_i : z_i = k\}, v)$ (Also $\sigma^2_k \sim \ldots$ and $\pi \sim \ldots$)

Example: Political Ideology as Latent Factor



Source: The Supreme Court Database by Lee Epstein, University of Southern California; Andrew D. Martin, Washington University in St. Louis; and Kevin Quinn, University of Michigan.

- Martin-Quinn scores: factor analysis-like model for Supreme Court justices, cases, and their votes on each case `https://mqscores.lsa.umich.edu/`
- Model is fit with Gibbs sampling, including conjugate normal updates.
- Established, well-validated measure of judicial ideology.

Bayesian inference software

- *Stan* uses autodifferentiation for posterior gradients for the Hamiltonian MCMC algorithm. This enables declarative, general-purpose Bayesian learning, often geared toward statistical data analysis applications (medium dimensionality, hierarchy, but no NNs). Very widely used.
  `https://pystan.readthedocs.io/en/latest/`
  `https://mc-stan.org/`
- Various other packages and efforts are out there to support probabilistic programming languages and automatic inference/learning for other classes of models and systems.