COMS30035, Machine learning: Probabilistic Graphical Models 5

James Cussens
james.cussens@bristol.ac.uk

Department of Computer Science, SCEEM
University of Bristol

October 18, 2020

Agenda

- The Metropolis-Hastings algorithm

How to get MCMC to work?

- At the end of the last lecture we had a clear goal: **given** a target probability distribution $p(\mathbf{z})$, **construct** a Markov chain $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(i)} \ldots$ such that $\lim_{i\to\infty} p(\mathbf{z}^{(i)}) = p(\mathbf{z})$.
- (For Bayesian machine learning the target distribution will be $P(\theta|D = d)$, the posterior distribution of the model parameters given the observed data.)
- One solution to this is the *Metropolis-Hastings* algorithm.

The Metropolis-Hastings (MH) algorithm

- We define a single transition probability distribution for a homogeneous Markov chain.
- Let the current state be $\mathbf{z}^{(\tau)}$. When using the MH algorithm sampling the next state happens in two stages:
  1. We generate a value $\mathbf{z}^*$ by sampling from a *proposal distribution* $q(\mathbf{z}|\mathbf{z}^{(\tau)})$.
  2. We then accept $\mathbf{z}^*$ as the new state with a certain *acceptance probability*. If we don't accept $\mathbf{z}^*$ then we 'stay where we are', so that $\mathbf{z}^{(\tau)}$ is both the old and new state.

The Metropolis-Hastings acceptance probability

Let $p(\mathbf{z})$ be the *target distribution*. The acceptance probability is: [Bis06, p. 541].

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{p(\mathbf{z}^*)q(\mathbf{z}^{(\tau)}|\mathbf{z}^*)}{p(\mathbf{z}^{(\tau)})q(\mathbf{z}^*|\mathbf{z}^{(\tau)})}\right) \quad (1)$$

- If $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z$ then we have $p(\mathbf{z}^*)/p(\mathbf{z}^{(\tau)}) = \tilde{p}(\mathbf{z}^*)/\tilde{p}(\mathbf{z}^{(\tau)})$, so we only need p up to normalisation. This is a big win!
- If the proposal distribution is symmetric then the 'q' terms cancel out: a special case known as the *Metropolis algorithm*.
- Note that for the Metropolis algorithm if $p(\mathbf{z}^*) \geq p(\mathbf{z}^{(\tau)})$ then we always accept and 'move' to $\mathbf{z}^*$.

Does Metropolis-Hastings (always) work?

.

- It can be shown [Bis06, p. 541] that the target distribution is an *invariant distribution of the Markov chain*: if the sequence of distributions $p(\mathbf{z}^{(i)})$ reaches the target distribution then it stays there.
- Also, typically the Markov chain does converge to the target distribution.
- The *rate* at which we converge to the target distribution is greatly influenced by the choice of proposal distribution.

MCMC in practice

- Straightforward Metropolis-Hastings is not the state-of-the-art in MCMC.
- *Probabilistic programming* systems like PyMC3 by default use more sophisticated MCMC algorithms (to avoid getting stuck).
- From the PyMC3 webpage: “PyMC3 allows you to write down models using an intuitive syntax to describe a data generating process. Fit your model using gradient-based MCMC algorithms like NUTS, …”.
- When using MCMC we (1) throw away early samples (‘burn-in’) and (2) ‘run independent chains’ to check for convergence.
- PyMC3 uses $\hat{R}$ (`r_hat`) to check for convergence; this value should be close to 1.

Linear regression with pyMC3

```
import pymc3 as pm

X, y = linear_training_data()
with pm.Model() as linear_model:
    weights = pm.Normal('weights', mu=0, sigma=1)
    noise = pm.Gamma('noise', alpha=2, beta=1)
    y_observed = pm.Normal('y_observed',
                    mu=X @ weights,
                    sigma=noise,
                    observed=y)

    prior = pm.sample_prior_predictive()
    posterior = pm.sample()
    posterior_pred = pm.sample_posterior_predictive(
                                                posterior)
```

Now do the quiz!

Yes, please do the quiz for this lecture on Blackboard!

Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.