CSC 2541: Bayesian Methods for Machine Learning

Radford M. Neal, University of Toronto, 2011

Lecture 2

Monte Carlo Methods

A very general approach to Bayesian computation — applicable even to very high-dimensional problems — is to obtain a *sample* of points from the posterior distribution, and use it to make *Monte Carlo* estimates.

A single sample point will contain values for all the unknown parameters, hyperparameters, latent variables, missing data values, etc. — everything not known, except what we don't care about or have integrated away analytically.

We use this sample to approximate expected values by averages over the sample points. For example, from K values, $\theta^{(1)}, \ldots \theta^{(K)}$, for a parameter, sampled from $P(\theta \,|\, \text{data})$, we can approximate the predictive probability that $Y = 1$ by

$$\begin{aligned} P(Y = 1 \mid \text{data}) &= \int P(Y = 1 \mid \theta)\, P(\theta \,|\, \text{data})\, d\theta \\ &\approx \frac{1}{K} \sum_{k=1}^{K} P(Y = 1 \mid \theta^{(k)}) \end{aligned}$$

If the $\theta^{(k)}$ values are independent, the approximation converges to the true value as $K \to \infty$, by the Law of Large Numbers.

Monte Carlo with Independent Points

Monte Carlo is simplest when we can directly sample K *independent* points from the distribution of interest.

Let's denote the probability/density function of interest as $\pi(x)$, and suppose that we are interested in the expectation of some function $a(x)$. Note that x is typically high dimensional.

The Monte Carlo estimate based on K sample points, $x^{(1)}, \ldots, x^{(K)}$, will be

$$\overline{a} \;=\; \frac{1}{K} \sum_{k=1}^{K} a(x^{(k)})$$

If the variance of $a(x)$ is finite, we can get an indication of the accuracy of this estimate from its standard error — an estimate of the standard deviation of $\overline{a}$ in imaginary repetitions of the estimation procedure. This standard error is

$$\text{S.E. } \overline{a} \;=\; \sqrt{s_a^2 / K}$$

where s_a^2 is the sample variance of a:

$$s_a^2 \;=\; \frac{1}{K-1} \sum_{k=1}^{K} \Big(a(x^{(k)}) - \overline{a} \Big)^2$$

Application: General Expectations for Conjugate Models

Efficient direct sampling of independent points from the posterior is usually possible only for models with conjugate priors. Typically, posterior means of parameters can be found analytically for such models, so Monte Carlo isn't necessary.

However, even for a conjugate model, the expectation of some complicated function of the parameters may be an integral that isn't analytically tractable. But a Monte Carlo estimate based on independent points can be found as long as the posterior can be efficiently sampled.

This is what really makes conjugate models tractable, even when the dimension of the parameter space is high.

Importance Sampling

When there is no efficient way to sample independently from $\pi(x)$ we can instead sample independently from some "similar" distribution, $\pi^*(x)$, and estimate the expectation of $a(x)$ by

$$\widehat{a}_{IS} = \frac{\sum_{k=1}^{K} a(x^{(k)}) \, \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})}}{\sum_{k=1}^{K} \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})}}$$

Note that we don't need the normalizing constants for π or π^*, since they will cancel in the ratio above.

As long as $\pi^*(x) > 0$ for all x where $\pi(x) > 0$, this converges to the expectation of $a(x)$ under π as $K \to \infty$. We can see this since

$$\frac{1}{K} \sum_{k=1}^{K} \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})} \to E_{\pi^*}\left[\frac{\pi(x)}{\pi^*(x)}\right] = \int \left[\frac{\pi(x)}{\pi^*(x)}\right] \pi^*(x)\, dx = 1$$

$$\frac{1}{K} \sum_{k=1}^{K} a(x^{(k)}) \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})} \to E_{\pi^*}\left[a(x) \, \frac{\pi(x)}{\pi^*(x)}\right] = E_{\pi}[a(x)]$$

Accuracy of Importance Sampling

Here's the importance sampling estimate again:

$$\widehat{a}_{IS} = \frac{\sum_{k=1}^{K} a(x^{(k)}) \, \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})}}{\sum_{k=1}^{K} \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})}}$$

If we know the normalizing constants for π or π^*, we could omit the denominator, since it converges to one. But that estimator is less accurate, so we shouldn't.

We can get a standard error for $\widehat{a}_{IS}$ by taking the square root of an estimate of its variance:

$$\mathrm{Var}(\widehat{a}_{IS}) \approx \frac{\sum_{k=1}^{K} \left(\frac{\pi(x^{(k)})}{\pi^*(x^{(k)})} \, (a(x^{(k)}) - \widehat{a}_{IS}) \right)^2}{\left[\sum_{k=1}^{K} \frac{\pi(x^{(k)})}{\pi^*(x^{(k)})} \right]^2}$$

This is discussed in my paper on "Annealed Importance Sampling".

Usefulness of Importance Sampling

The usefulness of importance sampling depends crucially on whether a good π^* can be found, that can be efficiently sampled, *and* leads to $\widehat{a}_{IS}$ being accurate.

Accuracy will be poor if $\pi^*(x)$ is very small in a region with non-negligible probability under π — then few points will be sampled from a region that actually is important to estimating $E_\pi(a(x))$.

Worse, it's possible that **no** points will be sampled from this region — then the estimate will be inaccurate, but the standard error obtained may not indicate that it is inaccurate.

But you can't just make π^* be very broad — then most points sampled will be wasted, with $\pi(x)$ being very small.

Direct use of importance sampling for Bayesian inference is usually practical only in moderate dimensions (eg, 10), and then only after significant fiddling to get a good π^* (eg, some heavy-tailed distribution located at the posterior mode).

Obtaining a Sample by Simulating a Markov Chain

When the posterior distribution is too complex to sample from directly, we can instead simulate a *Markov chain* that will converge (asymptotically) to the posterior distribution.

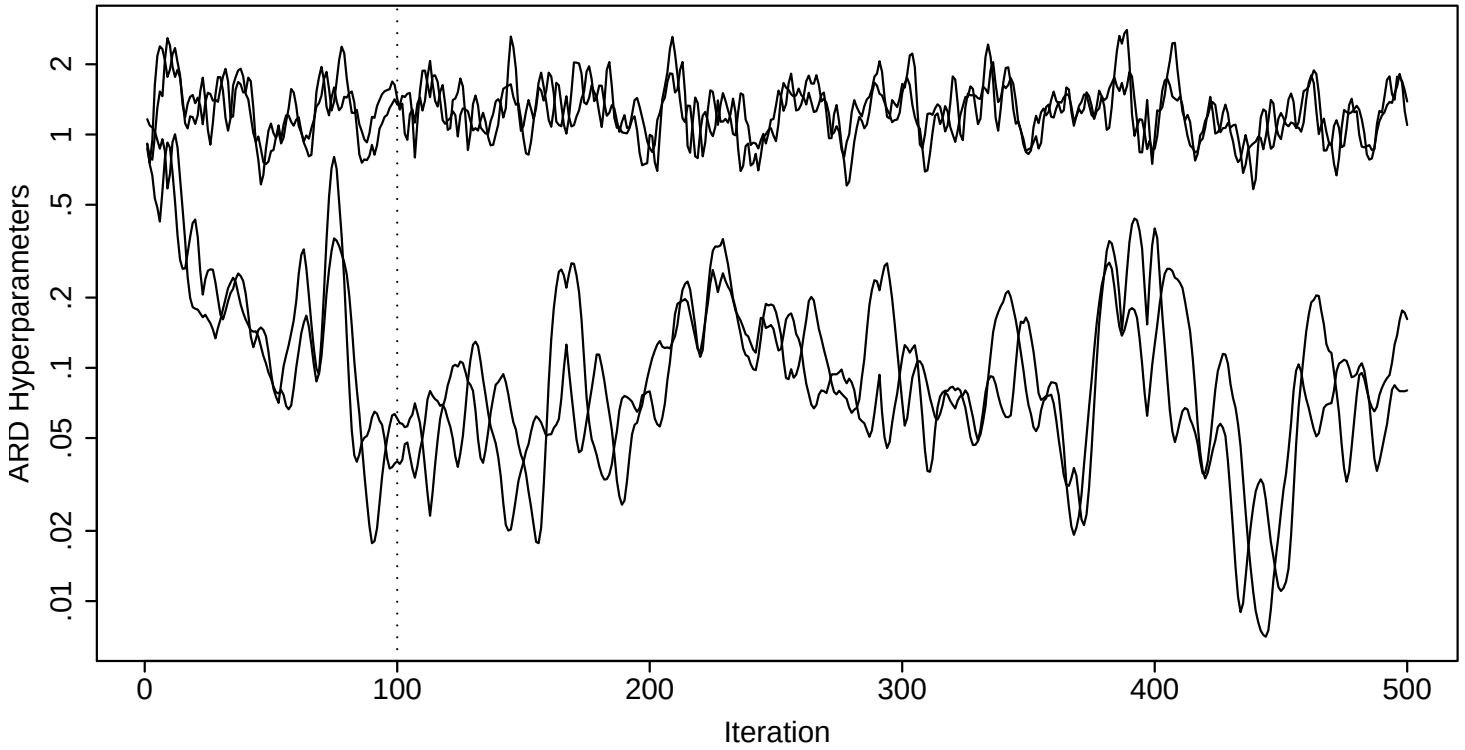
States from the latter portion of this Markov chain will come (asymptotically) from the posterior distribution, but they will be *dependent*.

We can still use these states to make Monte Carlo estimates, but we need to adjust the standard error to account for the dependence.

Finding such a Markov chain sounds hard, but fortunately there are general schemes that make this possible even for difficult problems. This *Markov chain Monte Carlo (MCMC)* approach is therefore very general. MCMC can also be very slow in some circumstances, but despite this, it is often the only viable approach to Bayesian inference using complex models.

An Example of Markov Chain Sampling

Here is a Markov chain run for a Gaussian process classification model, showing the four hyperparameters controlling the relevance of the four input variables:



The chain starts in a state that isn't typical of the posterior distribution, but by about iteration 100, the distribution seems to have stabilized. We would use iterations from there on to make predictions.

Note the high autocorrelation for the two hyperparameters with low values. Fortunately, the exact degree of irrelevance of largely irrelevant inputs isn't crucial.

Fundamental Requirements for Markov Chain Monte Carlo

Suppose we want to sample from a distribution $\pi(x)$ by simulating a Markov chain — eg, π could be the posterior, x the parameter vector. For notational simplicity, we'll assume here that x is discrete, but everything generalizes.

We need to find transition probabilities, $T(x, x')$, for the chain to move from state x to state x' that will lead to convergence to π.

A fundamental requirement for this is that the transitions *leave* π *invariant*:

$$\pi(x') \quad = \quad \sum_x \pi(x)\, T(x, x'), \quad \text{for all } x'$$

This says that if we ever reached the distribution π at some time in the simulation, the distribution for the next state would also be π.

We also need the chain to be *ergodic* — roughly speaking, it shouldn't get trapped in one part of the state space.

These two conditions are enough to guarantee that Monte Carlo estimates based on states from the chain converge to the correct expectations with respect to π.

Proving Invariance From Detailed Balance

Our first challenge is to find transition probabilities that leave π invariant.
One way is to show that the transitions satisfy a stronger condition known as *detailed balance*:

$$\pi(x)\,T(x,x') \;=\; \pi(x')\,T(x',x), \quad \text{for all } x \text{ and } x'$$

If this is true, the chain is also said to be *reversible* with respect to π.

It's easy to prove that detailed balance implies invariance:

$$\begin{aligned}\sum_x \pi(x)\,T(x,x') &= \sum_x \pi(x')\,T(x',x) \\ &= \pi(x')\sum_x T(x',x) \;=\; \pi(x')\end{aligned}$$

The converse is not true: *nonreversible* chains that leave π invariant are possible (and many are useful).

Combining Transitions

If the transitions $T_0(x, x')$ and $T_1(x, x')$ both leave π invariant, then so does any mixture of the two, define for some $\alpha \in [0, 1]$ by

$$T_\alpha(x, x') \quad = \quad (1 - \alpha)T_0(x, x') + \alpha T_1(x, x')$$

The transition defined by first applying T_0 and then applying T_1 will also leave π invariant. If we view transition probabilities as a matrix, this combined transition matrix is just $T_0 T_1$.

Two applications:

- We can combine several transitions each of which changes only part of the state (leaving the rest unchanged). As long as each part is changed by at least one of these transitions, the combined transition may be ergodic.
- We can combine several types of transitions in the hopes that at least one of them will work well. It may be that we need one type in one part of the state space, another type in another part.

The Metropolis Algorithm

The original MCMC method, applied to a statistical physics problem by Metropolis, *et al.* in 1953, is still widely used, because it is very widely applicable.

A *Metropolis algorithm* does a transition from state x to state x' as follows:

1) A "candidate", x^*, is proposed according to some probabilities $S(x, x^*)$, satisfying the symmetry condition, $S(x, x^*) = S(x^*, x)$.

2) This candidate, x^*, is accepted as the next state with probability

$$\min\Big[1,\ \pi(x^*)/\pi(x)\Big]$$

If x^* is accepted, then $x' = x^*$. If x^* is instead rejected, then $x' = x$.

Transitions defined in this way leave π invariant, since they satisfy detailed balance — for any x and x' with $x \neq x'$:

$$\begin{aligned}
\pi(x)\,T(x,x') &= \pi(x)\,S(x,x')\,\min\Big[1,\ \pi(x')/\pi(x)\Big] \\
&= S(x,x')\,\min\Big[\pi(x),\ \pi(x')\Big] \\
&= \pi(x')\,S(x',x)\,\min\Big[1,\ \pi(x)/\pi(x')\Big] \;=\; \pi(x')\,T(x',x)
\end{aligned}$$

More on the Metropolis Algorithm

Since π enters only in the ratio $\pi(x^*)/\pi(x)$, it's enough to be able to evaluate some function proportional to $\pi(x)$ — the normalizing constant cancels.

Each Metropolis update (after the first) requires only one evaluation of π, if the value of $\pi(x)$ was saved from the previous update.

The choice of proposal distribution, S, is important, since it determines whether the chain is ergodic, and if so, whether it converges rapidly, and moves around the distribution rapidly after convergence.

Random walk proposals are usually used, with the distribution of x^* being centred at the current state.

Two Types of Metropolis Algorithms

Multivariate Metropolis: Proposals change all components of x.

Single-variable Metropolis: Proposals change only one component of x. Updates for each component are applied in sequence (or for each update a component is chosen at random).

Single-variable updates are especially attractive when incremental computations allow $\pi(x')$ to be quickly found if $\pi(x)$ was previously computed, and x' differs from x in only one component. Then a sequence of updates for all components may be almost as fast as a single update of all components at once.

Proposals that change just a few components of x are also possible, of course, and may sometimes benefit from incremental computation.

Example: Logistic Regression

I'll demonstrate multivariate Metropolis on the posterior distribution for a logistic regression model of n independent observations, (x_i, y_i), with $x_i \in R^2$ and $y_i \in \{0, 1\}$:

$$P(y_i = 1 \,|\, x_i, \beta_0, \beta_1, \beta_2) \quad = \quad [1 + \exp(-\beta_0 - \beta_1 x_1 - \beta_2 x_2)]^{-1}$$

with independent priors for the parameters as follows:

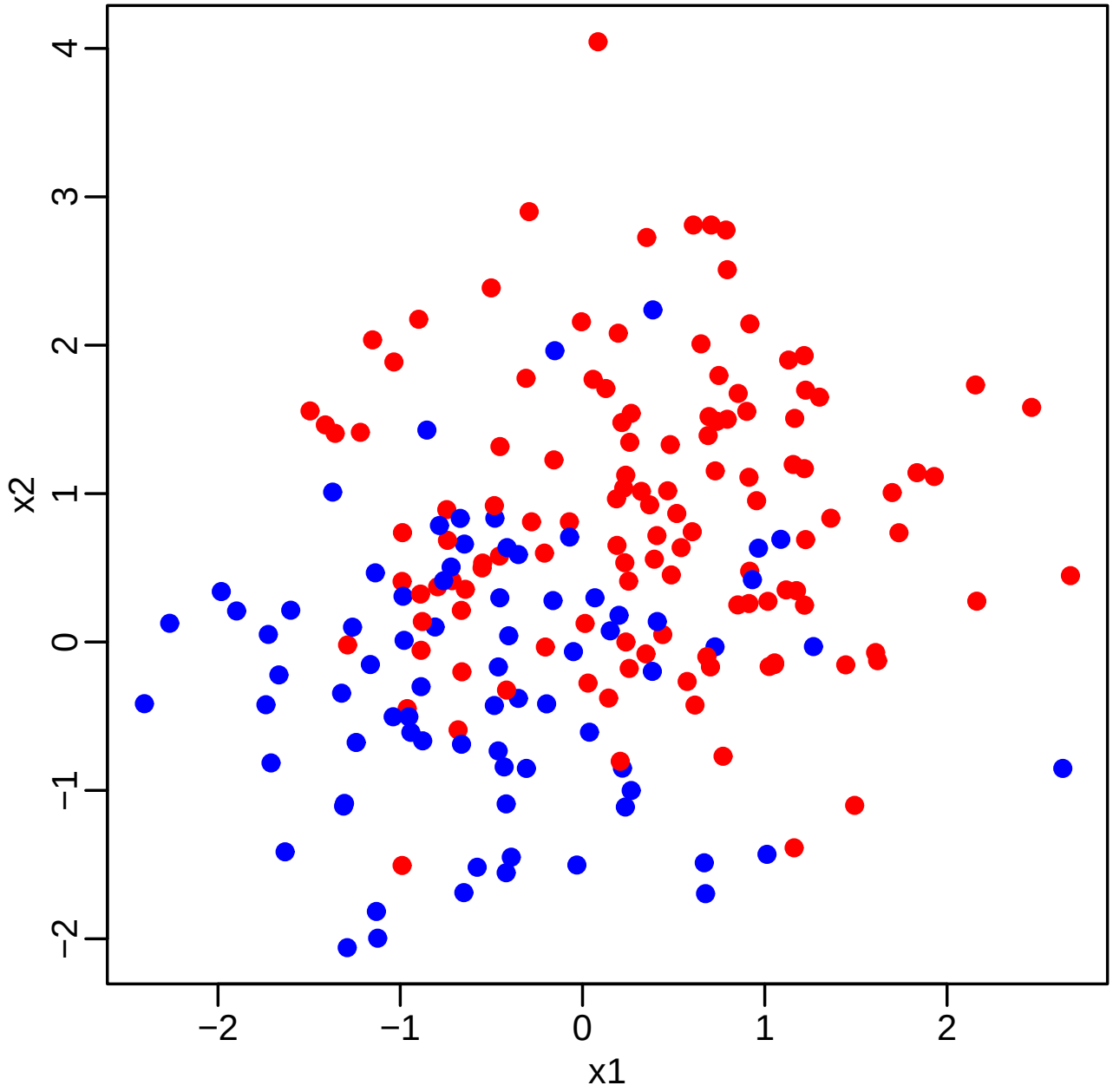
$$\begin{aligned} \beta_0 &\sim N(0, 10^2) \\ \beta_1 &\sim N(0, 1) \\ \beta_2 &\sim N(0, 1) \end{aligned}$$

This model does not have a conjugate prior, nor are there any low-dimensional sufficient statistics. (You need to look at all the observations to compute the likelihood.)

Otherwise, it's pretty easy though. The posterior is unimodal.

We'll use Metropolis to sample from the posterior distribution given 200 observations.

Data for the Logistic Regression Example



Metropolis for the Logistic Regression Example

I samped from the posterior distribution using a Metropolis algorithm with proposal distribution in which the proposed β_0, β_1, and β_2 are independently drawn from normal distributions with means equal to their current values and standard deviations of 0.3.

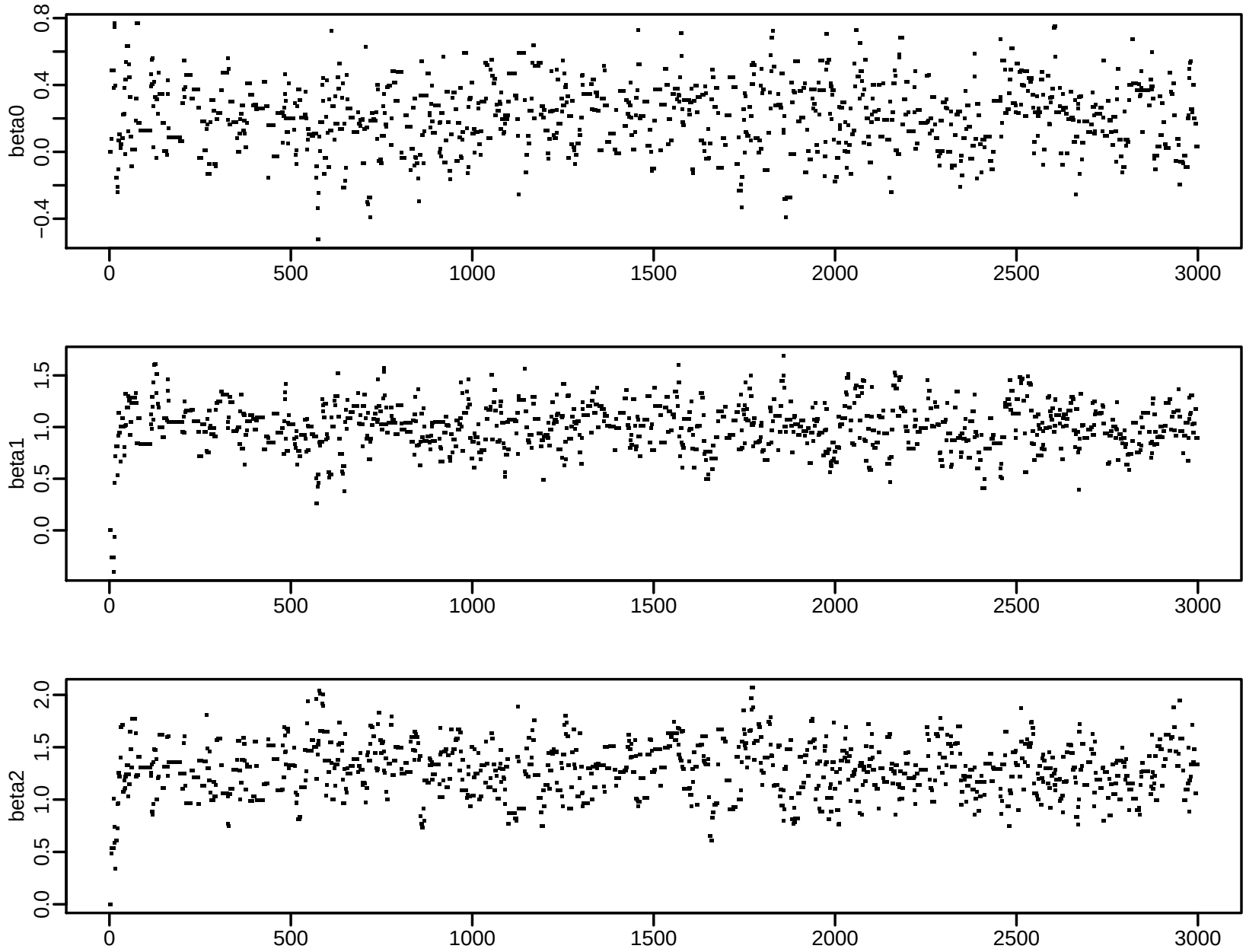
I did 3000 iterations. Trace plots are shown on the next slide.

The rejection rate for proposals was 71%

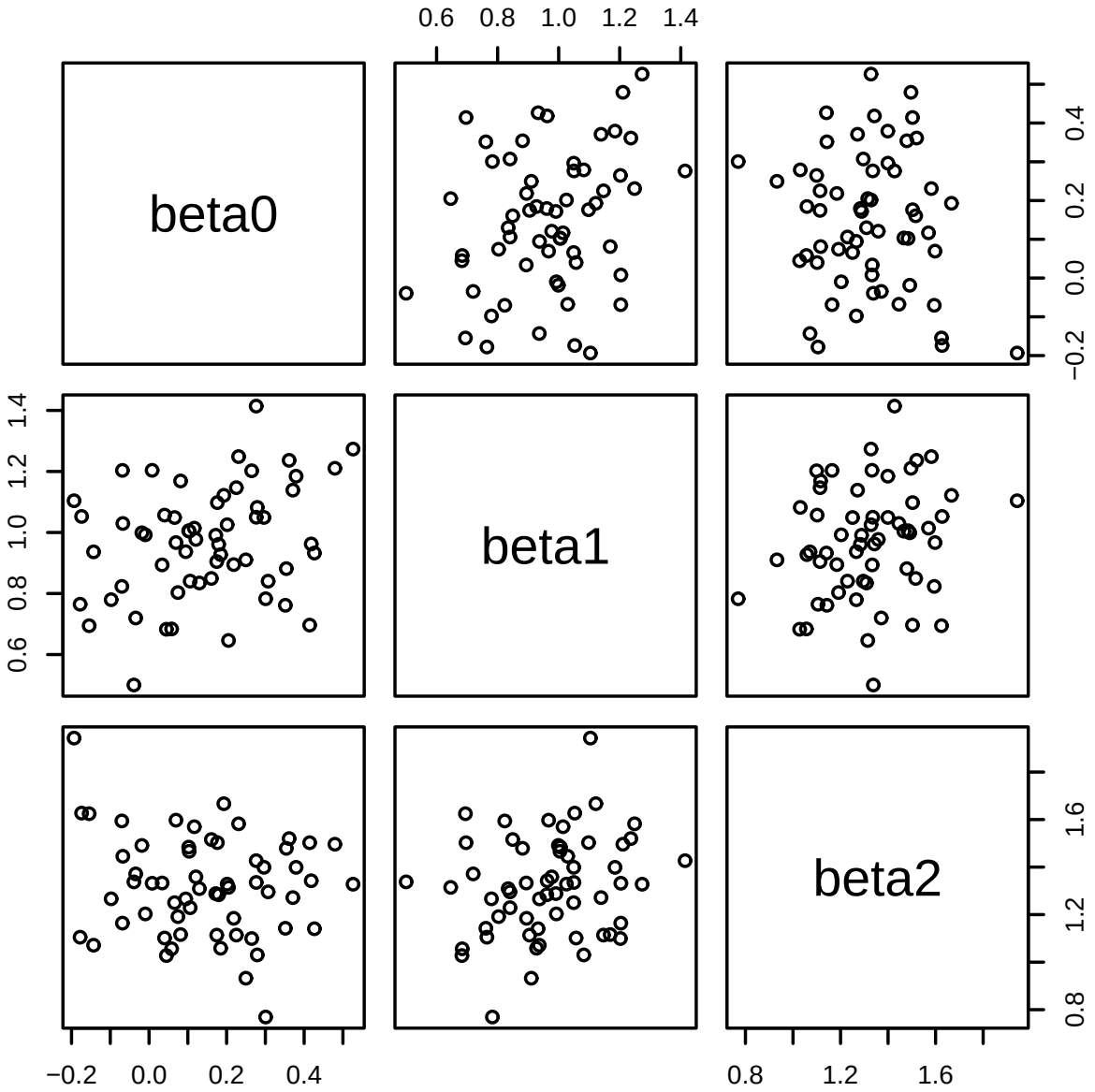
I discarded iterations before 100 as "burn in" — perhaps not from the right distribution because the chain might not have (almost) converged yet.

For the later plots, I took every 50th iteration from 100 on, so that there would be no duplicates from rejections (and so that the plot doesn't have too many points/lines). Such "thinning" is *not* necessary when computing numerical estimates of expectations (though it might sometimes be necessary to save memory).

Trace Plots of Metropolis for Logistic Regression Example



Scatterplots of Posterior for Logistic Regression Example



Posterior of Separators for Logistic Regression Example