Numerical integration

Joachim Vandekerckhove

Motivating application

In cognitive science (and science generally), we are often interested in the probability that a theory is true or the probability of values of parameters.

"Now that we have these data, what are the likely values of the generating parameters?"

Motivating application: Signal detection

What is the probability distribution of a, assuming that

$$\forall i : \begin{cases} \theta_i^f & = & \frac{f_i}{N} \\ \theta_i^h & = & \Phi\left(a + \Phi^{-1}\left(\theta_i^f\right)\right) \\ f_i & \sim & \text{Bin}\left(\theta_i^f, N\right) \\ h_i & \sim & \text{Bin}\left(\theta_i^h, S\right) \end{cases}$$

and given the data D, which consists of hits h_i, false alarms f_i, signal count S, and noise count N?

In other words, what is $p\left(a|D\right)$?

Motivating application: Signal detection

We can actually work out that distribution:

$$p\left(a|D\right) = \frac{p\left(D|a\right)p\left(a\right)}{p\left(D\right)} \propto p\left(D|a\right)p\left(a\right)$$

But now what? We may want to characterize this distribution:

- Get the mean of this distribution as a good guess for a

Motivating application: Signal detection

We can actually work out that distribution:

$$p\left(a|D\right) = \frac{p\left(D|a\right)p\left(a\right)}{p\left(D\right)} \propto p\left(D|a\right)p\left(a\right)$$

But now what? We may want to characterize this distribution:

- Get the mean of this distribution as a good guess for a
- Calculate the probability that it is in a certain range

Motivating application: Signal detection

We can actually work out that distribution:

$$p(a|D) = \frac{p(D|a)\,p(a)}{p(D)} \propto p(D|a)\,p(a)$$

But now what? We may want to characterize this distribution:

- Get the mean of this distribution as a good guess for a
- Calculate the probability that it is in a certain range
- Get its standard deviation for a measure of uncertainty

Motivating application: Signal detection

- Get the mean of this distribution as a good guess for a:

$$E(a|D) = \int_{-\infty}^{+\infty} a\, p(a|D)\, da$$

Motivating application: Signal detection

- Get the mean of this distribution as a good guess for a:

$$E(a|D) = \int_{-\infty}^{+\infty} a\, p(a|D)\; da$$

- Calculate the probability that it is in a certain range:

$$P(L < a < U|D) = \int_{L}^{U} p(a|D)\; da$$

Motivating application: Signal detection

- Get the mean of this distribution as a good guess for a:

$$E(a|D) = \int_{-\infty}^{+\infty} a\, p(a|D)\; da$$

- Calculate the probability that it is in a certain range:

$$P(L < a < U|D) = \int_{L}^{U} p(a|D)\; da$$

- Get its standard deviation for a measure of uncertainty:

$$S(a|D) = \sqrt{E(a^2) - [E(a)]^2}$$

Numerical integration methods

Some basic methods

- The trapezoid rule

Some basic methods

- The trapezoid rule
    - Divide the domain into small intervals

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$
  - Evaluate the curve at well-chosen quadrature points

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$
  - Evaluate the curve at well-chosen quadrature points
  - Approximate the integral with a weighted average

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$
  - Evaluate the curve at well-chosen quadrature points
  - Approximate the integral with a weighted average
- Monte Carlo methods

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$
  - Evaluate the curve at well-chosen quadrature points
  - Approximate the integral with a weighted average
- Monte Carlo methods
  - Draw random samples that fall under the curve

Some basic methods

- The trapezoid rule
  - Divide the domain into small intervals
  - Approximate the curve over each interval by a simple shape
  - Approximate the integral by adding areas of the simple shapes
- Gaussian quadrature
  - Only for curves of the form $f(x)\phi(x)$
  - Evaluate the curve at well-chosen quadrature points
  - Approximate the integral with a weighted average
- Monte Carlo methods
  - Draw random samples that fall under the curve
  - Characterize the curve with summary statistics of the sample

Monte Carlo methods

Numerical integration

A very convenient approximation to the expectation integral is

$$\begin{aligned} E(a|D) &= \int_{-\infty}^{+\infty} a\, p(a|D)\; da \\ &\approx \frac{1}{R}\sum_{r=1}^{R} a_r \text{ where } \forall r : a_r \overset{iid}{\sim} p(a|D) \end{aligned}$$

Numerical integration

A very convenient approximation to the expectation integral is

$$
\begin{aligned}
E(a|D) &= \int_{-\infty}^{+\infty} a\, p(a|D)\, da \\
&\approx \frac{1}{R} \sum_{r=1}^{R} a_r \text{ where } \forall r : a_r \overset{iid}{\sim} p(a|D)
\end{aligned}
$$

That is, the sample mean of a (large) random sample drawn from the distribution. Other summary statistics (like standard deviation) can be computed in a similar way. This property is known as ergodicity.

Numerical integration

A very convenient approximation to the expectation integral is

$$
\begin{aligned}
E(a|D) &= \int_{-\infty}^{+\infty} a\, p(a|D)\, da \\
&\approx \frac{1}{R} \sum_{r=1}^{R} a_r \text{ where } \forall r : a_r \overset{iid}{\sim} p(a|D)
\end{aligned}
$$

That is, the sample mean of a (large) random sample drawn from the distribution. Other summary statistics (like standard deviation) can be computed in a similar way. This property is known as ergodicity.

It follows that in order to characterize an arbitrary distribution, it suffices to be able to draw random samples from it.

Monte Carlo methods

- Sampling-based methods for describing distributions are called Monte Carlo methods and they are extremely useful.

Monte Carlo methods

- Sampling-based methods for describing distributions are called Monte Carlo methods and they are extremely useful.
- They are called Monte Carlo methods after the famous casino in Monaco where a relative of the method's creator, Stanislaw Ulam, enjoyed gambling.

Monte Carlo methods

- Sampling-based methods for describing distributions are called Monte Carlo methods and they are extremely useful.
- They are called Monte Carlo methods after the famous casino in Monaco where a relative of the method's creator, Stanislaw Ulam, enjoyed gambling.
- Their initial development in 1947 almost immediately followed the completion of ENIAC, the first general-purpose digital computer, in 1945.

Monte Carlo methods

- Sampling-based methods for describing distributions are called Monte Carlo methods and they are extremely useful.
- They are called Monte Carlo methods after the famous casino in Monaco where a relative of the method's creator, Stanislaw Ulam, enjoyed gambling.
- Their initial development in 1947 almost immediately followed the completion of ENIAC, the first general-purpose digital computer, in 1945.
- There are many MC methods, but the most common ones are Markov chain Monte Carlo (MCMC) methods.

Metropolis sampler

- A widely applicable algorithm is the Metropolis algorithm.

Metropolis sampler

- A widely applicable algorithm is the Metropolis algorithm.
- Named after Nicholas Metropolis, who created it together with John von Neumann while working on the Manhattan Project.

Metropolis sampler

- A widely applicable algorithm is the Metropolis algorithm.
- Named after Nicholas Metropolis, who created it together with John von Neumann while working on the Manhattan Project.
- In the algorithm, we will randomly generate candidate samples from some simple distribution, and then decide to accept or reject the candidate.

Metropolis sampler

- A widely applicable algorithm is the Metropolis algorithm.
- Named after Nicholas Metropolis, who created it together with John von Neumann while working on the Manhattan Project.
- In the algorithm, we will randomly generate candidate samples from some simple distribution, and then decide to accept or reject the candidate.
- Metropolis algorithms need some customization and fine-tuning to be most efficient.

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1 `Set` $i \leftarrow 1$ and `choose` sample size R

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1 `Set` $i \leftarrow 1$ and `choose` sample size R

2 `Choose`, arbitrarily, initial state $\theta^{(0)}$

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1 `Set` $i \leftarrow 1$ and `choose` sample size R

2 `Choose`, arbitrarily, initial state $\theta^{(0)}$

3 `Draw` a randomly selected proposal θ^c from $Q\left(\theta|\theta^{(i-1)}\right)$

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1. `Set` $i \leftarrow 1$ and `choose` sample size R
2. `Choose`, arbitrarily, initial state $\theta^{(0)}$
3. `Draw` a randomly selected proposal θ^{c} from $Q\left(\theta|\theta^{(i-1)}\right)$
4. `Compute` the acceptance ratio $\alpha = \frac{f(\theta^c)}{f\left(\theta^{(i-1)}\right)} = \frac{p(\theta^c|D)}{p\left(\theta^{(i-1)}|D\right)}$

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1. Set $i \leftarrow 1$ and choose sample size R
2. Choose, arbitrarily, initial state $\theta^{(0)}$
3. Draw a randomly selected proposal θ^c from $Q\left(\theta|\theta^{(i-1)}\right)$
4. Compute the acceptance ratio $\alpha = \frac{f(\theta^c)}{f\left(\theta^{(i-1)}\right)} = \frac{p(\theta^c|D)}{p\left(\theta^{(i-1)}|D\right)}$
5. Draw a randomly selected uniform variate u from $U(0,1)$. If $\alpha > u$, set state $\theta^{(i)} \leftarrow \theta^c$, otherwise set state $\theta^{(i)} \leftarrow \theta^{(i-1)}$

Metropolis sampler: Pseudocode

Given a target function $f(\theta) \propto p(\theta|D)$ and a symmetric candidate generating distribution $Q(x|y) = Q(y|x)$, a Metropolis sampling algorithm proceeds as follows:

1 Set $i \leftarrow 1$ and choose sample size R

2 Choose, arbitrarily, initial state $\theta^{(0)}$

3 Draw a randomly selected proposal θ^c from $Q\left(\theta|\theta^{(i-1)}\right)$

4 Compute the acceptance ratio $\alpha = \frac{f(\theta^c)}{f\left(\theta^{(i-1)}\right)} = \frac{p(\theta^c|D)}{p\left(\theta^{(i-1)}|D\right)}$

5 Draw a randomly selected uniform variate u from $U(0,1)$. If $\alpha > u$, set state $\theta^{(i)} \leftarrow \theta^c$, otherwise set state $\theta^{(i)} \leftarrow \theta^{(i-1)}$

6 Set $i \leftarrow i+1$. If $i \leq R$, return to Step 3, otherwise halt

Metropolis sampler: Common choices

- R will often be large, like 5,000.

Metropolis sampler: Common choices

- R will often be large, like 5,000.
- $Q()$ will often be something that's easy to sample, like a normal distribution.

Metropolis sampler: Common choices

- R will often be large, like 5,000.
- $Q()$ will often be something that's easy to sample, like a normal distribution.
- $p(\theta|D)$ will often be our posterior distribution.

Metropolis sampler: Common choices

- R will often be large, like 5,000.
- $Q()$ will often be something that's easy to sample, like a normal distribution.
- $p(\theta|D)$ will often be our posterior distribution.
- Often, for computational stability, we will deal with $\log(p(\theta|D))$, in which case we come the log of the acceptance probability $\log(\alpha) = \log(p(\theta^c|D)) - \log\left(p(\theta^{(i-1)}|D)\right)$ and compare it to the log of a uniform variate, $\log(u)$.

Metropolis sampler: Adaptation

- Using a normal distribution as the candidate generating distribution brings an advantage: we can choose the width of it so that the CGD envelops the target distribution tightly.

Metropolis sampler: Adaptation

- Using a normal distribution as the candidate generating distribution brings an advantage: we can choose the width of it so that the CGD envelops the target distribution tightly.
- That way, more samples can be accepted and the algorithm can be more efficient.

Metropolis sampler: Adaptation

- Using a normal distribution as the candidate generating distribution brings an advantage: we can choose the width of it so that the CGD envelops the target distribution tightly.
- That way, more samples can be accepted and the algorithm can be more efficient.
- We will tune the sampler so that it accepts approximately 40% of all proposed samples.

Metropolis sampler: Adaptation

- Using a normal distribution as the candidate generating distribution brings an advantage: we can choose the width of it so that the CGD envelops the target distribution tightly.
- That way, more samples can be accepted and the algorithm can be more efficient.
- We will tune the sampler so that it accepts approximately 40% of all proposed samples.
- During the adaptation phase, we will "warm up" the algorithm but the samples drawn during this phase are not yet samples from the target distribution, so we discard them.

Metropolis sampler: Adaptation

- The adaptation phase goes like this:

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
  1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
    1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted
    2. If that fraction is too low, decrease the standard deviation because it's jumping around too wildly

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
  1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted
  2. If that fraction is too low, decrease the standard deviation because it's jumping around too wildly
  3. If it is too high, increase the standard deviation because it's climbing a slope

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
  1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted
  2. If that fraction is too low, decrease the standard deviation because it's jumping around too wildly
  3. If it is too high, increase the standard deviation because it's climbing a slope
  4. Repeat this a few times to get an acceptance rate close to 40%

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
  1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted
  2. If that fraction is too low, decrease the standard deviation because it's jumping around too wildly
  3. If it is too high, increase the standard deviation because it's climbing a slope
  4. Repeat this a few times to get an acceptance rate close to 40%
- A simple rule to update the standard deviation is

$$\sigma_{\text{new}} = \sigma_{\text{old}} \times \left(\frac{r_{\text{target}}}{r_k} \right)^{1.1}$$

Metropolis sampler: Adaptation

- The adaptation phase goes like this:
  1. Run the sampler for some iterations n_k and keep track of the acceptance rate r_k: the fraction of samples that gets accepted
  2. If that fraction is too low, decrease the standard deviation because it's jumping around too wildly
  3. If it is too high, increase the standard deviation because it's climbing a slope
  4. Repeat this a few times to get an acceptance rate close to 40%
- A simple rule to update the standard deviation is

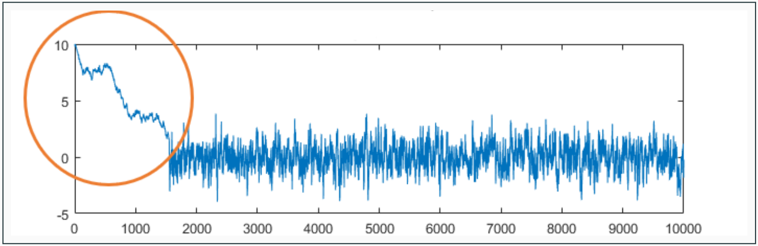
$$\sigma_{\text{new}} = \sigma_{\text{old}} \times \left(\frac{r_{\text{target}}}{r_k} \right)^{1.1}$$

- How much to fine-tune depends on the specific case.

Metropolis sampler: Post-processing

- We have to make sure the chain has converged to a stationary sampling state before using the samples for inference.

Trace plot

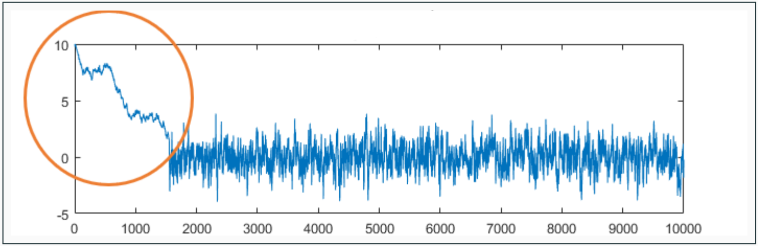


Metropolis sampler: Post-processing

- We have to make sure the chain has converged to a stationary sampling state before using the samples for inference.
- Often, we will discard a number of initial samples known as the burn-in:

$$\hat{a} = \frac{1}{R - B} \sum_{i=B+1}^{R} a^{(i)}$$

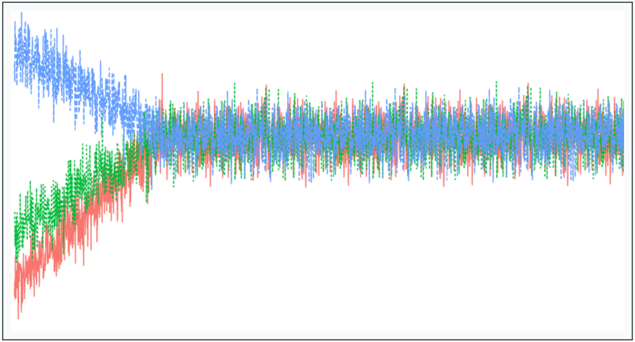
Trace plot



Metropolis sampler: Diagnostics

- We usually also repeat the procedure a few times with different values for $\theta^{(0)}$ to ensure that the algorithm converges to the same stationary distribution.

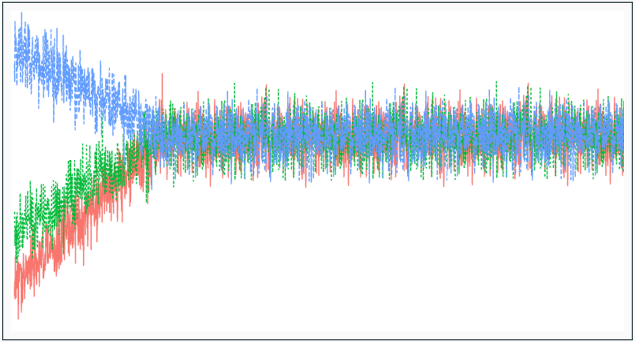
Trace plot



Metropolis sampler: Diagnostics

- We usually also repeat the procedure a few times with different values for $\theta^{(0)}$ to ensure that the algorithm converges to the same stationary distribution.
- Several convergence statistics exist, with Geweke's and Gelman's $\hat{R}$ being the most popular.

Trace plot



Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
    1. Draw samples

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
    1. Draw samples
    2. Generate a synthetic data set from each sample

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
    1. Draw samples
    2. Generate a synthetic data set from each sample
    3. Calculate a summary statistic on each synthetic data set and visualize the distribution

Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
  1. Draw samples
  2. Generate a synthetic data set from each sample
  3. Calculate a summary statistic on each synthetic data set and visualize the distribution
  4. Calculate the same statistic on the real data and show its position in the distribution

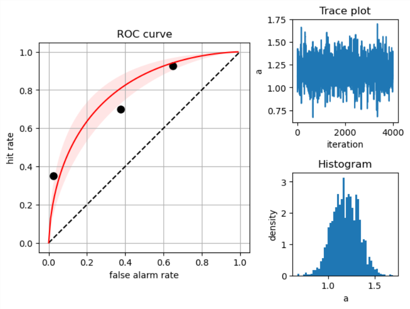
Metropolis sampler: Inference

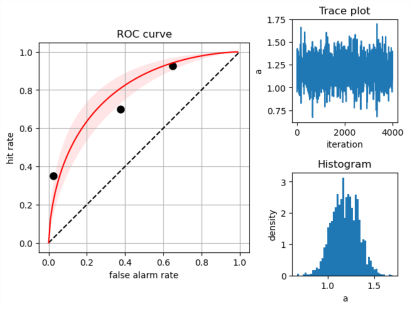
- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
    1. Draw samples
    2. Generate a synthetic data set from each sample
    3. Calculate a summary statistic on each synthetic data set and visualize the distribution
    4. Calculate the same statistic on the real data and show its position in the distribution
- Figures can also be summary statistics!

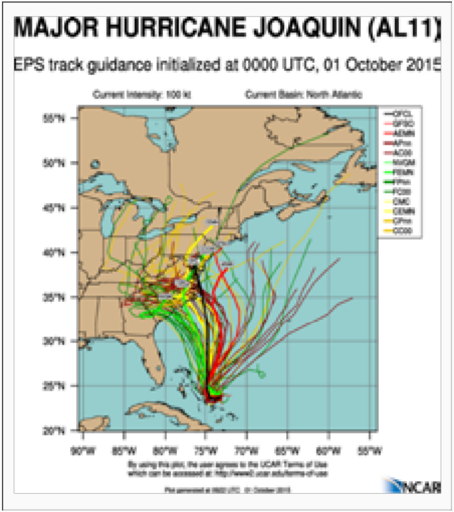
Metropolis sampler: Inference

- Finally, we calculate the quantities we're interested in using.
- The quantities can be simple summaries of the samples in our MCMC chain (mean, median, standard deviation, probabilities in intervals, quantiles...).
- However, they can also be complex functions of the samples, such as posterior predictives – entire simulated data sets:
    1. Draw samples
    2. Generate a synthetic data set from each sample
    3. Calculate a summary statistic on each synthetic data set and visualize the distribution
    4. Calculate the same statistic on the real data and show its position in the distribution
- Figures can also be summary statistics!
    - You could draw a curve to visualize your data and then draw a distribution of synthetic curves using your sampled parameters

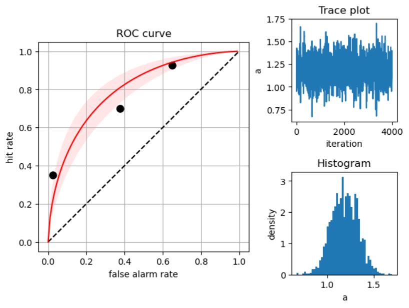
Metropolis sampler: Inference

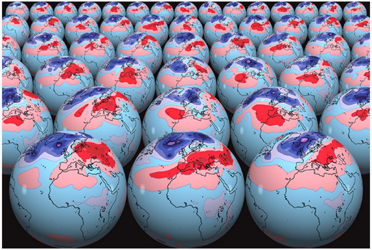


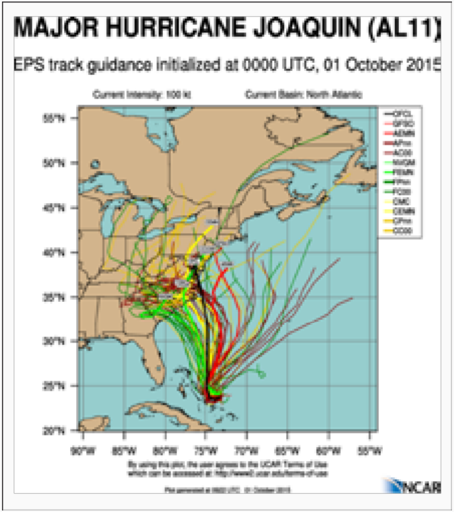

ROC curve
hit rate
false alarm rate
Trace plot
iteration
a
Histogram
density
a


MAJOR HURRICANE JOAQUIN (AL11)
EPS track guidance initialized at 0000 UTC, 01 October 2015
Current Basin: North Atlantic
NCAR

Metropolis sampler: Inference







Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$

Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$
2. Adapt: Run K blocks of n_k samples to find a good standard deviation σ for the CGD

Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$
2. Adapt: Run K blocks of n_k samples to find a good standard deviation σ for the CGD
3. Sample: Run a block of R samples to get a chain of R states

Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$
2. Adapt: Run K blocks of n_k samples to find a good standard deviation σ for the CGD
3. Sample: Run a block of R samples to get a chain of R states
4. Post-process: Burn samples before convergence

Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$
2. Adapt: Run K blocks of n_k samples to find a good standard deviation σ for the CGD
3. Sample: Run a block of R samples to get a chain of R states
4. Post-process: Burn samples before convergence
5. Diagnose: Visualize chains and calculate convergence statistics

Metropolis sampler: Overview

Typical steps of a Metropolis analysis are thus:

1. Set up: Select target function $f(\theta)$ and initial state $\theta^{(0)}$
2. Adapt: Run K blocks of n_k samples to find a good standard deviation σ for the CGD
3. Sample: Run a block of R samples to get a chain of R states
4. Post-process: Burn samples before convergence
5. Diagnose: Visualize chains and calculate convergence statistics
6. Inference: Calculate summary statistics of interest

Numerical integration

- The Metropolis sampler is but one example of a large family of Monte Carlo algorithms.

Numerical integration

- The Metropolis sampler is but one example of a large family of Monte Carlo algorithms.
- Numerical integration is an active area of research, with new methods and algorithms being developed to address specific types of problems and improve the accuracy and efficiency of existing techniques.

Numerical integration

- The Metropolis sampler is but one example of a large family of Monte Carlo algorithms.
- Numerical integration is an active area of research, with new methods and algorithms being developed to address specific types of problems and improve the accuracy and efficiency of existing techniques.
- The development of computers and numerical algorithms in the 20th century greatly expanded the range of problems that could be solved using numerical integration. They are partly responsible for the Bayesian revolution in the 21st century.

Numerical integration

Joachim Vandekerckhove