LECTURE 7: Monte Carlo Sampling and Integration

- So far we solved linear and nonlinear least squares using MAP/MLE solution and use Laplace approximation at MAP for the posterior
- This is exact for linear least squares, but not for nonlinear dependence on parameters: in general the posteriors are analytically untractable

LECTURE 7: Monte Carlo Sampling and Integration

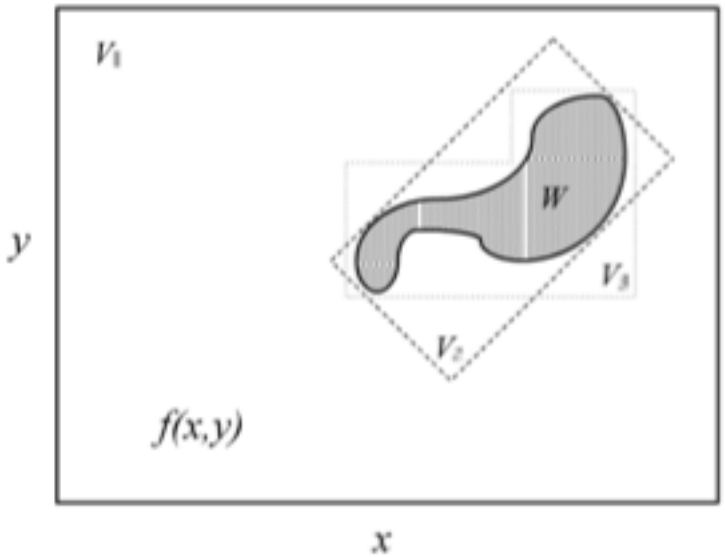
- Typically we want the posterior distribution of 1 or 2 (3 is already hard to visualize) parameters marginalizing over all the rest
- Marginalization = integration
- In low dimensions quadrature methods (Lecture 1) may be useful
- In high dimensions we resort to sampling methods

Simple Monte Carlo Integration

- We wish to compute N-dim integral

$$\int f dV \approx V\langle f \rangle \pm V\{(\langle f^2 \rangle - \langle f \rangle^2)/N\}^{1/2}$$

- $\langle f \rangle = \Sigma_{i=1}^{N} f(x_i)/N \qquad \langle f^2 \rangle = \Sigma_{i=1}^{N} f^2(x_i)/N$
- Error scales as $N^{-1/2}$
- Suppose the region W is complicated: then we evaluate it over a larger region V that is simpler, setting $f = 0$ wherever it falls out of the region
- Choosing the region V well makes a lot of difference in computing time



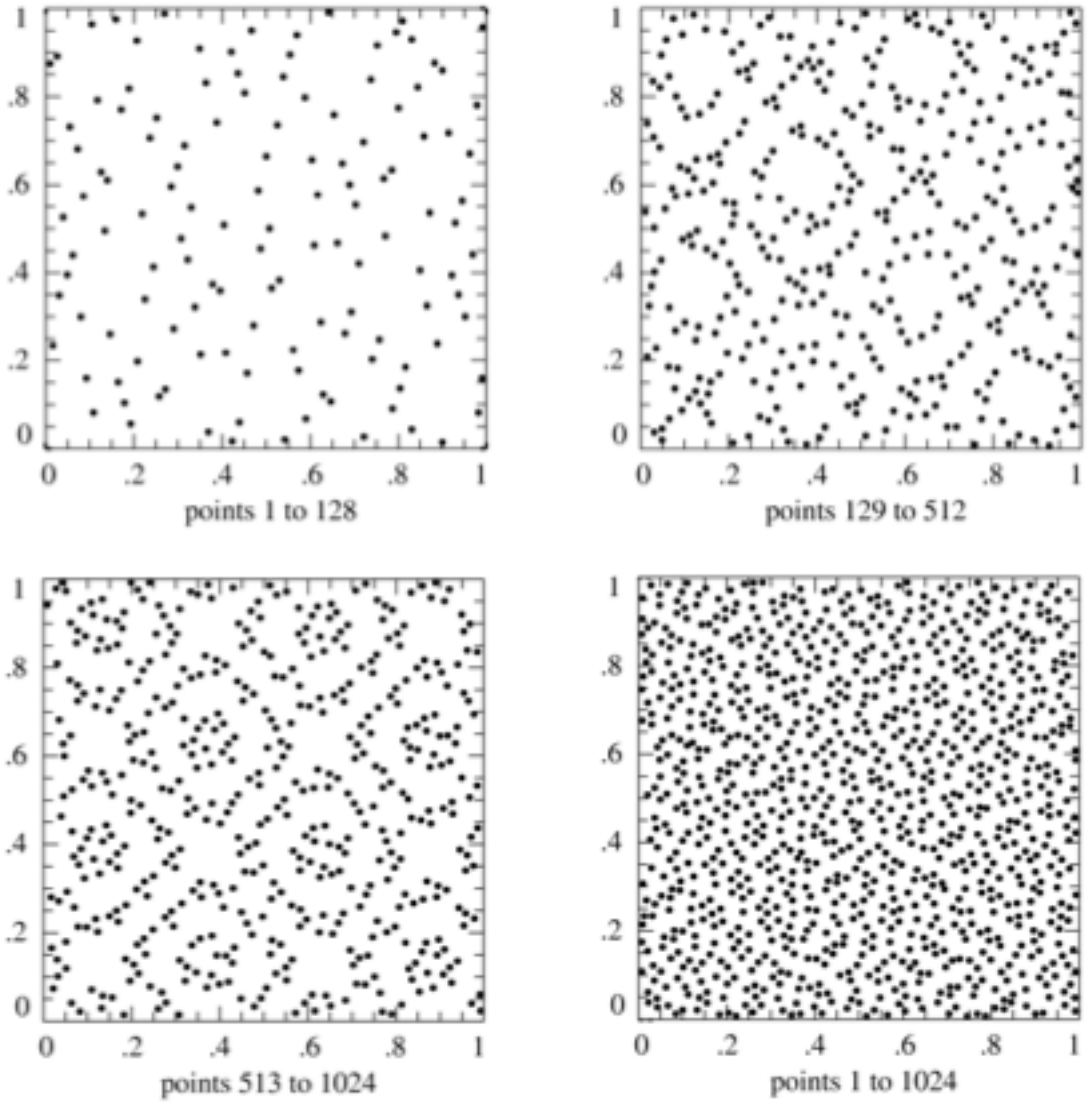
Simple Monte Carlo for Posteriors

- $p(\theta|d)=p(d|\theta)p(\theta)/p(d)$
- We can sample θ from $p(\theta)$, apply weight $p(d|\theta)$ and store the samples and their weights
- Posterior is simply average of these points including the weights: this automatically takes care of normalization (Bayes factor $p(d)$)
- To do this we must be able to sample θ from $p(\theta)$. For uniform prior as in previous slide this is easy, but we can also sample from more complicated distributions
- we use (pseudo) random number generators
- In filter theory this method is called particle filter
- Weight $p(d|\theta)$ is called importance sampling
- One often resamples the points to remove those with low weights

Random Number Generation

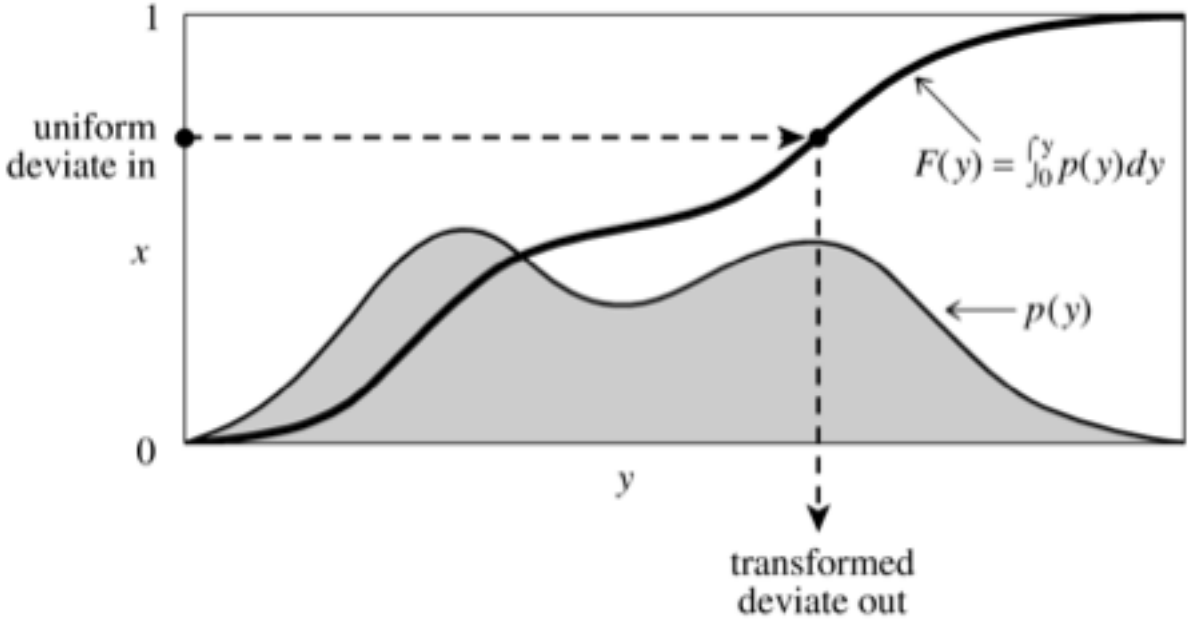
- No computer generated sequence is truly random, but pseudo-random (random enough for our purposes)
- This can be a good thing: the same random seed produces the same sequence of pseudo-random numbers, which means we can repeat the analysis if needed
- Sometimes we want to suppress sampling noise, so we use the same random number seed
- There is also a concept of quasi-random numbers, which attempt to make the Monte Carlo integrals converge faster than $N^{-1/2}$: e.g. Sobol sequence
- Most random number generators draw a uniform random number between 0 and 1
- We need to convert this to other $p(\theta)$

Sobol Sequence: Quasi-random in 2d



Inverse Transform Method

- Sometimes we can use transform of variable: $p(x)dx = p(y)dy$ hence $p(y) = p(x)|dx/dy|$
- Example: $y = -ln(x)$, $p(y) = |dx/dy| = e^{-y}$
- General: integrate $dx/dy = p(y)$, get $x = F(y) = \int p(y)dy$ hence $y = F^{-1}(x)$
- Works if we can compute $F^{-1}(x)$



Box-Muller Method for Gaussian

- We take 2 uniform variables x_1, x_2

$$p(y_1, y_2, \ldots)dy_1 dy_2 \ldots = p(x_1, x_2, \ldots)\left|\frac{\partial(x_1, x_2, \ldots)}{\partial(y_1, y_2, \ldots)}\right| dy_1 dy_2 \ldots$$

$$x_1 = \exp\left[-\frac{1}{2}(y_1^2 + y_2^2)\right] \qquad y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2$$

$$x_2 = \frac{1}{2\pi}\arctan\frac{y_2}{y_1} \qquad y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2$$

- Compute Jacobian

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}}e^{-y_1^2/2}\right]\left[\frac{1}{\sqrt{2\pi}}e^{-y_2^2/2}\right]$$

- We obtained 2 independent Gaussian distributed variables y_1, y_2

Multi-variate Gaussian

- We want to generate vector x from

$$N(\mathbf{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2}\det(\boldsymbol{\Sigma})^{1/2}}\exp[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\cdot\boldsymbol{\Sigma}^{-1}\cdot(\mathbf{x}-\boldsymbol{\mu})]$$

- We Cholesky decompose $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ and generate $\mathbf{y}$ as a gaussian with unit variance, then use $\mathbf{x} = \mathbf{L}\mathbf{y} + \boldsymbol{\mu}$

- Proof:

$$\langle \mathbf{y}\otimes\mathbf{y}\rangle = \mathbf{1} \qquad \langle(\mathbf{x}-\boldsymbol{\mu})\otimes(\mathbf{x}-\boldsymbol{\mu})\rangle = \langle(\mathbf{L}\mathbf{y})\otimes(\mathbf{L}\mathbf{y})\rangle$$
$$= \left\langle \mathbf{L}(\mathbf{y}\otimes\mathbf{y})\mathbf{L}^{T}\right\rangle = \mathbf{L}\,\langle\mathbf{y}\otimes\mathbf{y}\rangle\,\mathbf{L}^{T}$$
$$= \mathbf{L}\mathbf{L}^{T} = \boldsymbol{\Sigma}$$

Decorrelating Random Variables

- We discussed this in previous contexts, i.e. how to take a square root of a matrix. We can define $\mathbf{y} = \mathbf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu})$

$$\begin{aligned}\langle \mathbf{y} \otimes \mathbf{y} \rangle &= \langle (\mathbf{L}^{-1}[\mathbf{x} - \boldsymbol{\mu}]) \otimes (\mathbf{L}^{-1}[\mathbf{x} - \boldsymbol{\mu}]) \rangle \\ &= \mathbf{L}^{-1} \langle (\mathbf{x} - \boldsymbol{\mu}) \otimes (\mathbf{x} - \boldsymbol{\mu}) \rangle \mathbf{L}^{-1T} \\ &= \mathbf{L}^{-1} \boldsymbol{\Sigma} \mathbf{L}^{-1T} = \mathbf{L}^{-1} \mathbf{L}\mathbf{L}^T \mathbf{L}^{-1T} = \mathbf{1}\end{aligned}$$

- As we discussed this is not unique: any rotation by orthogonal matrix will also work:

$$\mathbf{K}^T\mathbf{K} = \mathbf{K}\mathbf{K}^T = \mathbf{1}$$

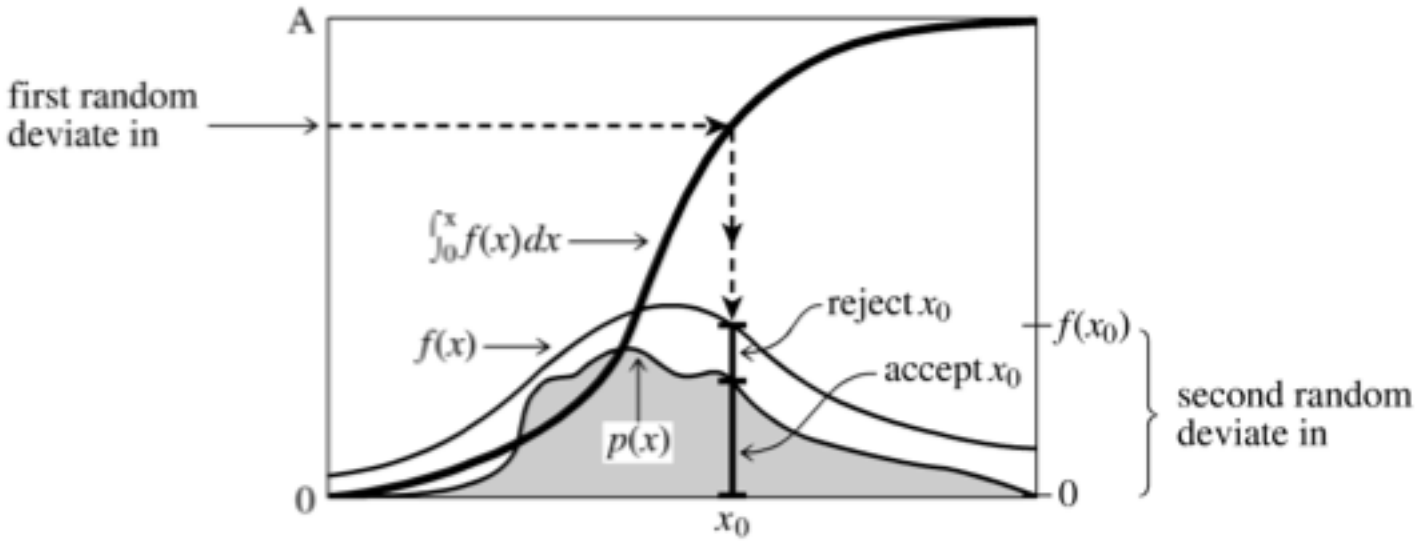
$$\langle (\mathbf{K}\mathbf{y}) \otimes (\mathbf{K}\mathbf{y}) \rangle = \mathbf{K} \langle \mathbf{y} \otimes \mathbf{y} \rangle \mathbf{K}^T = \mathbf{K}\mathbf{K}^T = \mathbf{1}$$

- Or we can diagonalize

$$\boldsymbol{\Sigma} = \mathbf{V}\mathrm{diag}(\sigma_i^2)\mathbf{V}^T$$

Rejection Sampling

- Find a function $f(x)$ for which $f(x) > p(x)$ and which is invertible
- Pick uniform random number y between 0 and A, determine $x_0 = F^{-1}(y)$
- Pick another random between 0 and 1 and accept x_0 with probability $p(x_0)/f(x_0)$



Rejection Sampling for Posteriors

- Same idea: we want to sample $p(\theta|d)$, we have $g(\theta)$ that is integrable and invertible and M where $Mg(\theta) > p(\theta|d)$
- We can sample θ from $g(\theta)$ first, then accept it with probability $p(\theta|d)/Mg(\theta)$

MC Integration over Infinite Intervals: Sampling from Non-uniform Distribution

- Goal: estimate $F_Y(y) = P(Y \leq y) = E\left[I_{(-\infty,y)}(Y)\right]$ where $Y \sim N(0,1)$:

$$F(Y \leq y) = \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = \int_{-\infty}^{\infty} h(t) \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

where $h(t) = 1$ if $t < y$ and $h(t) = 0$ if $t \geq y$

- Draw an iid sample $Y_1, \ldots, Y_N$ from a $N(0,1)$, then the estimator is

$$\hat{I} = N^{-1} \sum_{i=1}^{N} h(Y_i) = \frac{\#\text{ draws } < y}{N}$$

Importance Sampling Integration

- We wish to evaluate $I = \int h(y) f(y) dy$
- We rewrite it as sampling over g

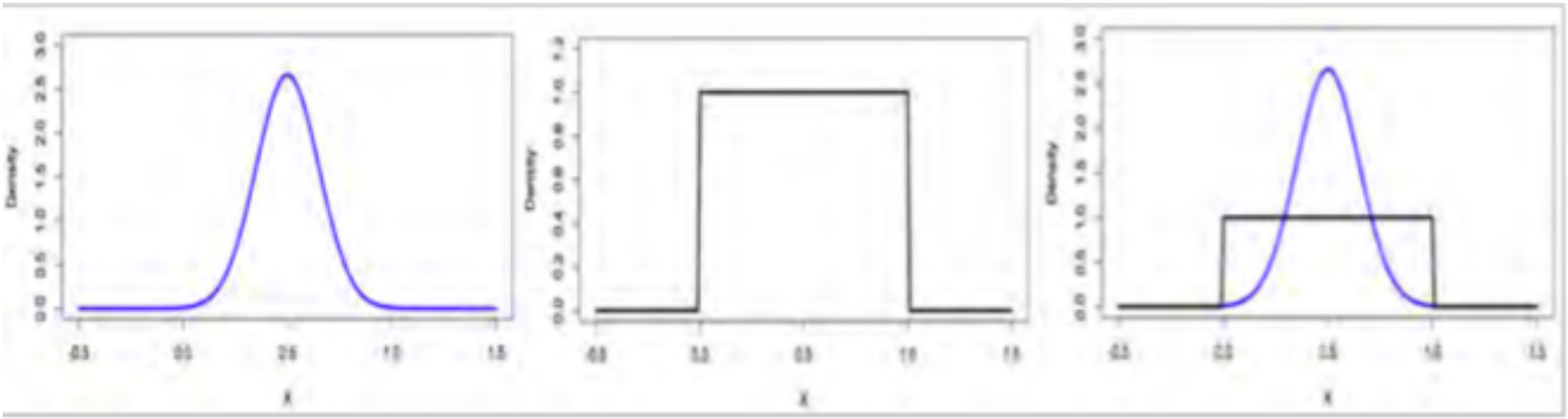
$$I = \int h(y) f(y) dy = \int h(y) \frac{f(y)}{g(y)} g(y) dy = \int \frac{h(y) f(y)}{g(y)} g(y) dy$$

$$I = E_f [h(Y)] = \int \frac{h(y) f(y)}{g(y)} g(y) dy = E_g \left[\frac{h(Y) f(Y)}{g(Y)} \right]$$

$$\hat{I} = N^{-1} \sum_{i=1}^{N} \frac{h(Y_i) f(Y_i)}{g(Y_i)} \longrightarrow E_g \left[\frac{h(Y) f(Y)}{g(Y)} \right] = I$$

Importance Sampling Integration

- Useful if we use MC to integrate a function whose value is very large or divergent, but integral is finite: there will be large fluctuations in the region where the function is divergent. We can change that with weights, placing more points there
- Useful if we cannot sample from *p(t)*: we sample from another distribution and reweight (generalization of rejection sampling)



Importance Sampling for Posteriors

- Suppose we have θ^s samples of $g(\theta)$ and we want to evaluate expectation values of $h(\theta)$ against $q(\theta|y)$:

$$\mathrm{E}(h(\theta|y)) = \frac{\int h(\theta) q(\theta|y) d\theta}{\int q(\theta|y) d\theta} = \frac{\int [h(\theta) q(\theta|y)/g(\theta)]\, g(\theta) d\theta}{\int [q(\theta|y)/g(\theta)]\, g(\theta) d\theta},$$

- We can approximate this with $\frac{\frac{1}{S}\sum_{s=1}^{S} h(\theta^s) w(\theta^s)}{\frac{1}{S}\sum_{s=1}^{S} w(\theta^s)}$,

  using importance weights $w(\theta^s) = \frac{q(\theta^s|y)}{g(\theta^s)}$

- Note that the weights can be > 1 and the method is not very useful when the values >> 1
- Effective sample size $S_{\mathrm{eff}} = \frac{1}{\sum_{s=1}^{S} (\tilde{w}(\theta^s))^2}$
- Importance sampling useful for mild variations of posterior (see Project 1): we have a data measurement which gives some posterior. Then data change. We can reuse the old posterior.

Markov Chain Monte Carlo Simulations

- Importance/rejection/simple MC are not very efficient in high dimensions unless the proposal function is very close to the true posterior: this results in a lot of weights being very small
- We want a method that selects the samples more efficiently, yet still guarantees convergence to a posterior
- MCMC generates a sequence of samples, where probability of drawing a new sample θ^t depends only on the last drawn sample θ^{t-1} and not on previous samples (Markov chain)
- This process is described with the transition distribution $T_t(\theta^t|\theta^{t-1})$

Detailed Balance and Equilibrium

- T satisifes detailed balance if $p^*(x)T(x,x') = p^*(x')T(x',x)$. This says the transition is reversible
- A distribution $p(x)$ is stationary if it is left unchanged under T: $p^*(x) = \Sigma_{x'} T(x',x)p^*(x')$
- If T satisfies detailed balance then the resulting distribution is stationary and given by the target distribution p': $\Sigma_{x'} p^*(x')T(x',x) = \Sigma_{x'} p^*(x)T(x,x') = p^*(x)$
- This only holds under ergodicity assumption: we must be able to reach any state in a finite number of steps from any initial state (e.g. no state can have zero prior…)

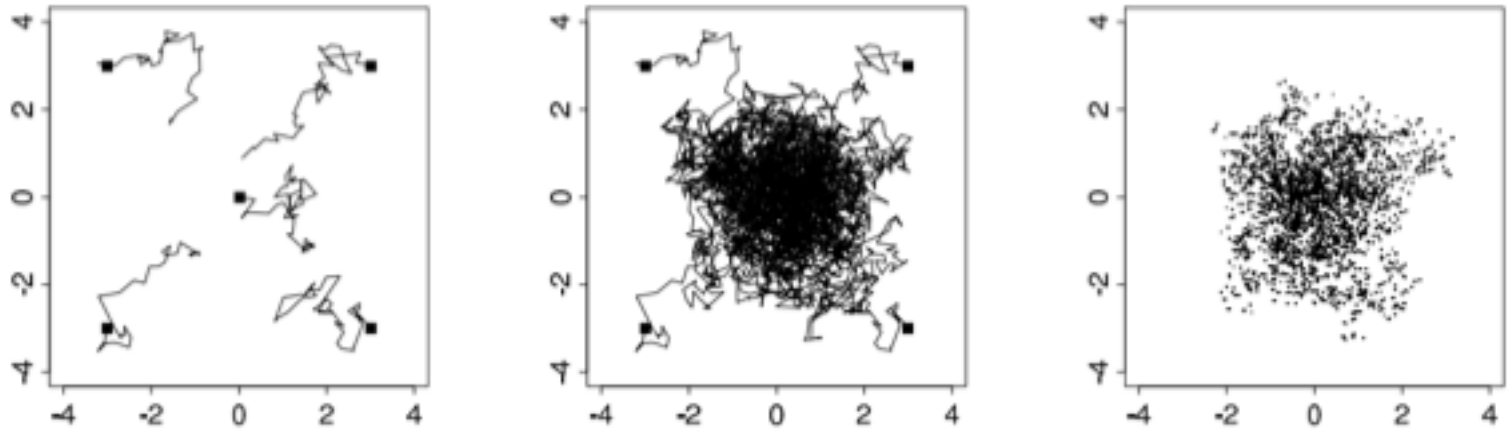
Metropolis Algorithm

- Origins: 1950's and glorious history of Los Alamos computing: Monte Carlo (von Neumann, Ulam, Metropolis), simulated annealing, MCMC. History has been kind to Metropolis, e.g.
- MCMC 1953 paper: posed by Teller, solved by M. Rosenbluth, coded by his wife Arianna, with apparently no contribution from Metropolis. Its origins go back to Fermi and Ulam.
- Algorithm: we sample from a proposal distribution $J_t(\theta^*|\theta^{t-1})$. The distribution is symmetric $J_t(\theta^a|\theta^b) = J_t(\theta^b|\theta^a)$
- Compute $r = p(\theta^*|d)/p(\theta^{t-1}|d)$
- If r > 1 always accept θ^*, if $r < 1$ accept θ^* with probability r, otherwise $\theta^t = \theta^{t-1}$
- we need to sample from J, so this must be simple. Example is $J_t(\theta^*|\theta^{t-1}) = \text{normal}(\theta^{t-1}, \sigma)$ with some σ such that it leads to an acceptance rate of order 25-50%.

Proof and Generalization to Metropolis-Hastings

- To prove it we must show it satisfies detailed balance
- Consider two point θ^a and θ^b drawn from p, $p(\theta^b|d) > p(\theta^a|d)$
- $p(\theta^{t-1}=\theta^a, \theta^t=\theta^b) = p(\theta^a|d)J_t(\theta^b|\theta^a)$ has acceptance 1
- $p(\theta^{t-1}=\theta^b, \theta^t=\theta^a) = p(\theta^b|d)J_t(\theta^a|\theta^b)$ $[p(\theta^a|d)/p(\theta^b|d)]$ where the last term (green) comes from accepting transition with probability $r = p(\theta^a|d)/p(\theta^b|d)$
- This gives $p(\theta^{t-1}=\theta^a, \theta^t=\theta^b) = p(\theta^t=\theta^a, \theta^{t-1}=\theta^b)$ since J_t is symmetric: detailed balance holds and $p(\theta|d)$ is the resulting stationary distribution
- Metropolis-Hastings generalizes this to non-symmetric J_t: $r = [p(\theta^*|d)/J_t(\theta^*|\theta^{t-1})] /[p(\theta^{t-1}|d)/ J_t(\theta^{t-1}|\theta^*)]$

How Does It Work?



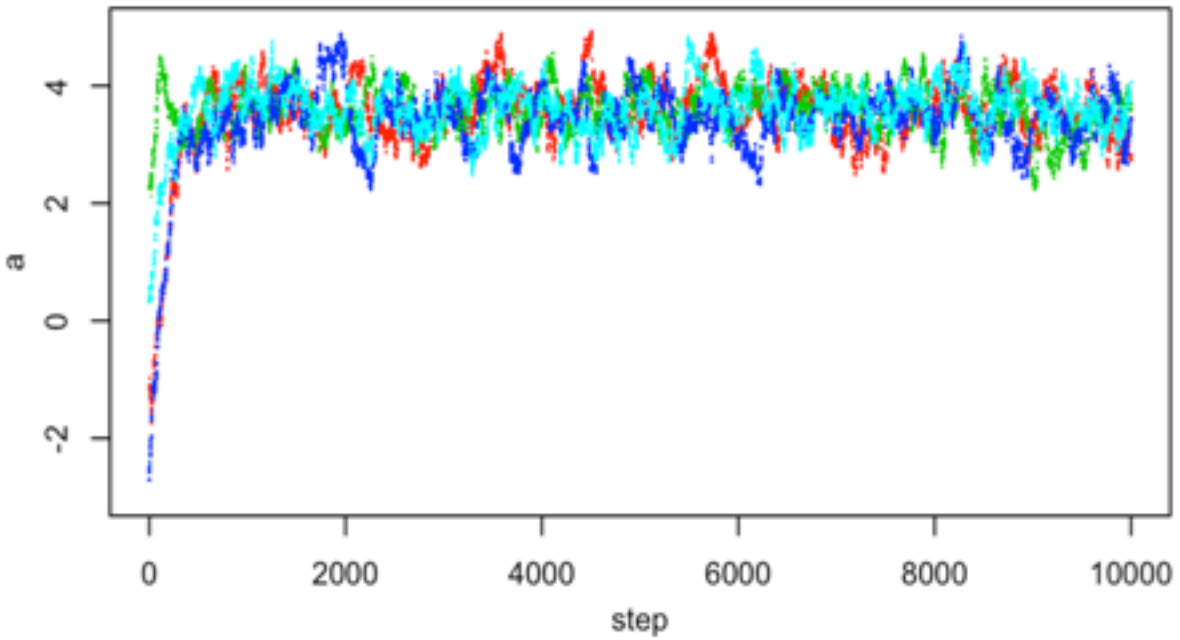
- Chains have burn in phase: initially chains are not sampling stationary distribution. Must check for convergence
- Chain events are correlated: one computes the correlation length N and thins the chain by keeping every *N*-th element
- It is advisable to have several chains with different initial conditions

Statistical Mechanics with MCMC

- In stat. mech. probability of a state with energy E_i is $p(E_i) = e^{-E_i/T}/Z$, where partition function $Z = \Sigma_i e^{-E_i/T}$ (we set k_b=1)
- Average of $\langle X \rangle = \Sigma_i p(E_i) X_i$ but this cannot be computed like this because $e^{-E_i/T}$ is small for most states: we want to importance sample with weight $w = p(E_i)$
- But we need Z to do $p(E)$, and it is easier to generate a sequence of samples using MCMC samples over all states i
- This is because Z cancels out in transition rule: $T_{ij}/T_{ji} = p(E_j)/p(E_i) = e^{-(E_j-E_i)/T}$
- To implement this we accept all transitions where energy goes down, and accept those where energy goes up with a probability $e^{-(E_j-E_i)/T}$ (we assume discrete states)
- Note that by doing this we simulate state transitions in nature

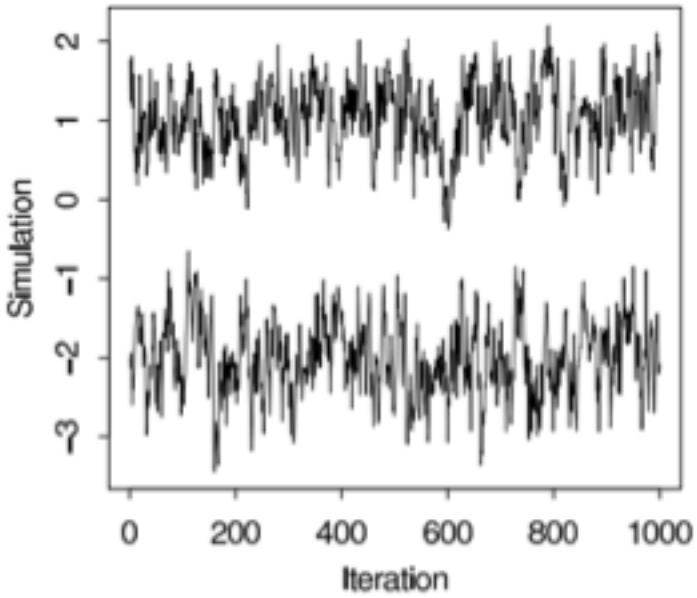
Convergence Tests

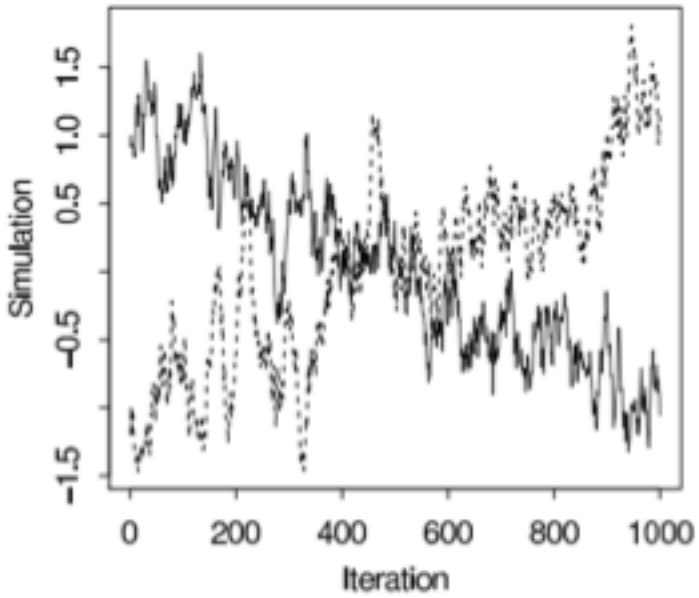
- We must perform convergence tests, starting with the visual inspection of the chain values as a function of t for all the parameters. Why? A chain may not have converged to the true posterior. For example, it may got stuck on a local minimum. Starting close to the global minimum is useful, for which optimization methods (Lecture 6) may provide a useful starting point.
- 1st step: discard burn-in phase



Convergence Tests

- Chains may have converged but not to the same parameter value
- Chains may not have converged (not stationary) but happen to be at the same place





Convergence Tests: Gelman-Rubin Statistic

- Tests similarity of independent chains converging to the same distribution
- If the chains are all having small fluctuations in a parameter within the chain, but fluctuations across the chains are large we have not converged
- Assume we have $j = 1, \ldots, m$ chains of length n: compute variance of q across chains by computing

  $$B = n(m-1)^{-1} \Sigma_{j=1}^{m} (<\theta_j> - <\theta>)^2$$

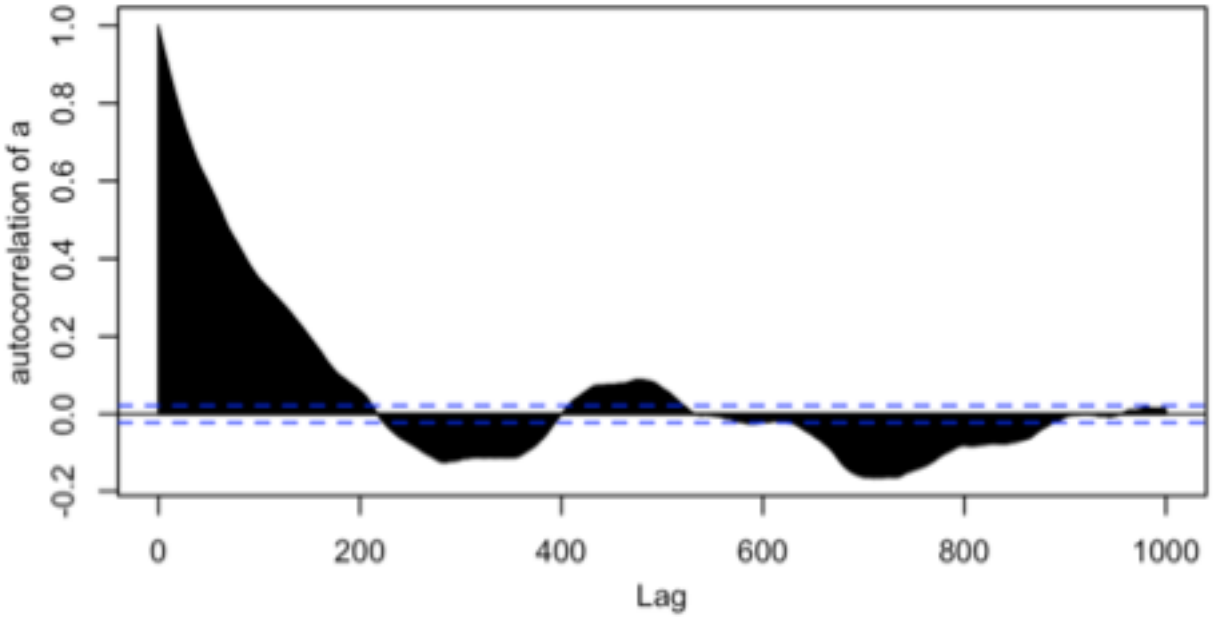
- Compute individual chain variance $W = m^{-1} \Sigma_{j=1}^{m} s_j^2$, where s_j^2 is the variance of q within chain j, $s_j^2 = \Sigma_{i=1}^{n} (<\theta_{ij}> - <\theta_j>)^2$
- Overall variance estimate is $V = W(n-1)/n + B/n$.
- Gelman Rubin statistic $R = (V/W)^{1/2}$. We like it to be close to 1 (e.g. $R < 1.1$)

Correlation Length

- Chain elements are not independent, and can be thinned without any loss
- Auto-correlation of a chain as a function of lag k is defined as

$$\rho_k = \frac{\sum_{i=1}^{n-k}(\theta_i - \bar{\theta})(\theta_{i+k} - \bar{\theta})}{\sum_{i=1}^{n-k}(\theta_i - \bar{\theta})^2} = \frac{\mathrm{Cov}_i(\theta_i, \theta_{i+k})}{\mathrm{Var}(\theta)}$$

- We can thin by 200!



More Efficient Samplers

- Metropolis MCMC remains popular, but can be quite inefficient if the proposal density does not match well the posterior, such as in the case of correlations between variables etc.
- We have also seen that the correlation length can be long, so effective number of samples is reduced and the number of posterior evaluations increases
- Often typical number is $10^6 - 10^7$: that is a lot of computer time, specially if the likelihood evaluation is expensive (in most cases because model evaluation is expensive)

How to make Metropolis efficient?

- Proposal function should be as close to posterior as possible. If this is a multi-variate gaussian with covariance S then it should be drawn from $J_t(\theta^*|\theta^{t-1}) = N(\theta^*|\theta^{t-1}, c^2\Sigma)$ where $c^2 = 2.4/d^{0.5}$. This gives acceptance rate of order 0.23-0.44 (depending on d).
- We can use initial chain to train Σ and then switch to this proposal, removing the first part (since it does not satisfy detailed balance).
- If the posterior is non-Gaussian we can try to convert to a Gaussian using a nonlinear transformation (e.g. Box-Cox transforms)
- One also needs to convert the prior via Jacobian!

Gibbs Sampler

- Here one samples each variable q_i at a time subject to conditional posterior $p(\theta_i|\ \theta_{-j})$, $\theta_{-j}= \theta_{j\neq i}$
- Why should this be better than sampling all θ_i at once? In general it is not. But if the conditional posterior can be evaluated analytically then it can be efficient. This happens when it is a conditionally conjugate distribution. If this happens then Gibbs sampler is the method of choice.
- Only works for conditional conjugate distributions that we know how to sample from.

Remember this slide from Lecture 3

- Conjugate prior: where posterior takes the same form as the prior
- Example: beta distribution is conjugate to binomial (HW 2)
- Can be interpreted as additional data
- For Gaussian with known σ : $p(\theta) \propto \exp\left(-\frac{1}{2\tau_0^2}(\theta-\mu_0)^2\right)$
- Posterior: $p(\theta|y) \propto \exp\left(-\frac{1}{2}\left(\frac{(y-\theta)^2}{\sigma^2}+\frac{(\theta-\mu_0)^2}{\tau_0^2}\right)\right)$
- Completing the square: $p(\theta|y) \propto \exp\left(-\frac{1}{2\tau_1^2}(\theta-\mu_1)^2\right)$

$$\mu_1 = \frac{\frac{1}{\tau_0^2}\mu_0 + \frac{1}{\sigma^2}y}{\frac{1}{\tau_0^2}+\frac{1}{\sigma^2}} \quad \text{and} \quad \frac{1}{\tau_1^2} = \frac{1}{\tau_0^2}+\frac{1}{\sigma^2}$$

Conditionally Conjugate Example for Gaussian Mean

- We observe y_1, y_2 drawn from a bivariate gaussian with unknown mean μ_1 and μ_2, but known covariance matrix with unity variance and correlation coefficient ρ.

$$\begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}\Big| y \sim \mathrm{N}\left(\begin{pmatrix}y_1\\ y_2\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right)$$

- For Gibbs sampling we need to compute marginals, which involves the usual gaussian integrals, which brings in Hessian matrix C^{-1}

$$\begin{aligned}\theta_1|\theta_2, y &\sim \mathrm{N}(y_1 + \rho(\theta_2 - y_2), 1-\rho^2)\\ \theta_2|\theta_1, y &\sim \mathrm{N}(y_2 + \rho(\theta_1 - y_1), 1-\rho^2)\end{aligned}$$

- We alternately sample from these two.

Gibbs Sampler has acceptance $r = 1$

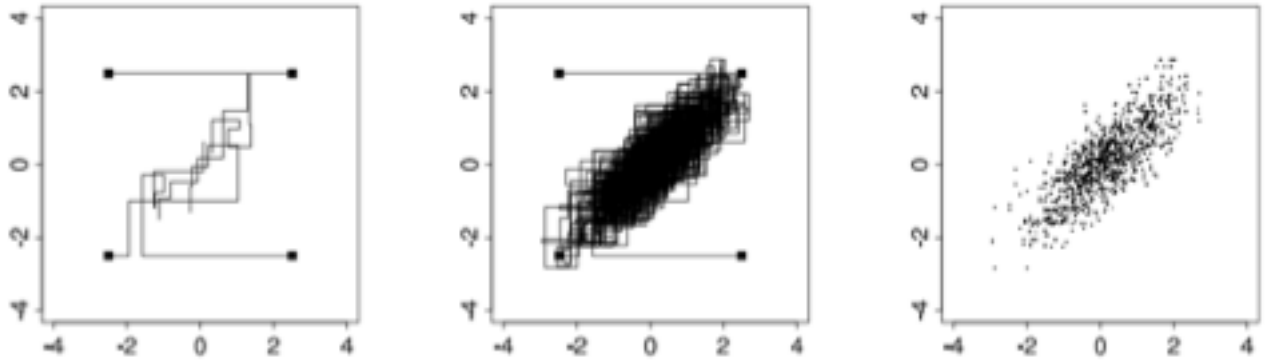
- It can be viewed as Metropolis-Hastings with acceptance rate = 1:

$$J_{j,t}^{\rm Gibbs}(\theta^*|\theta^{t-1}) = \begin{cases} p(\theta_j^*|\theta_{-j}^{t-1}, y) & \text{if } \theta_{-j}^* = \theta_{-j}^{t-1} \\ 0 & \text{otherwise.} \end{cases}$$

- $$r = \frac{p(\theta^*|y)/J_{j,t}^{\rm Gibbs}(\theta^*|\theta^{t-1})}{p(\theta^{t-1}|y)/J_{j,t}^{\rm Gibbs}(\theta^{t-1}|\theta^*)}$$
- $$= \frac{p(\theta^*|y)/p(\theta_j^*|\theta_{-j}^{t-1}, y)}{p(\theta^{t-1}|y)/p(\theta_j^{t-1}|\theta_{-j}^{t-1}, y)}$$
$$= \frac{p(\theta_{-j}^{t-1}|y)}{p(\theta_{-j}^{t-1}|y)}$$
$$\equiv 1,$$

Use $p(x|y)p(y) = p(y|x)p(x)$

$\theta_{-j}^* = \theta_{-j}^{t-1}$



- See BUGS/JAGS packages for Gibbs sampling

Population Monte Carlo

- Here we go back to the idea of importance sampling: instead of doing MCMC and worrying about detailed balance, we can draw samples from any population $q(\theta)$ and weight them by $p(\theta|d)/q(\theta)$
- This can only work if $q(\theta)$ is close to the posterior, otherwise some samples will have very large weight. PMC attempts to achieve this by iterating on $q(\theta)$ towards the posterior $p(\theta|d)$.
- Often this is done with a Gaussian mixture, starting with the Laplace approximation as one component and randomly displaced and stretched Gaussians as the other components.

Other MC Samplers

- Affine invariant sampling (Goodman-Weare) uses an ensemble of walkers, with updates for each walker based on positions of all other walkers: the chains are not independent! It often fails to converge if starting very far from the minimum. See emcee package.
- Another method that works for multi-modal distributions is DNest4, using a sequence of constrained prior distributions, where the constraint requires the likelihood to be above some value l. Also works for evidence calculations (Bayes factor).
- Multinest and Polychord are also methods that works for multi-modal distributions and evidence.
- Yet another method for multi-modal distributions is simulated annealing

Simulated Annealing for Optimization

- How do we reach global minimum in the presence of many local minima? Kirkpatrick (1985) proposed we mimic a physical system: in nature energy minimum is reached when T goes to 0. This is because at T=0 only ground state will be occupied, with $e^{-E_i/T} = 0$ for $E_i > 0$, and $e^{-E_i/T} = 1$ for $E_i = 0$
- However, if we cool too fast we get trapped in a local minimum. It has been known for centuries to glass makers that fast cooling of glass traps impurities (local minimum), while slow cooling (annealing) avoids this.

Simulated Annealing for Optimization

- To implement this for optimization we can modify the optimization function as $p^{1/T}$ and vary T, starting high and reaching 1 at the end.
- We also need a stochastic component, accepting a transition to another neighboring state even if the loss function increases. This can be modeled after MCMC with $r = p(\theta^*|d)/p(\theta^{t-1}|d)$, accepting with probability r if $r < 1$ and probability 1 otherwise.
- Optimal choice of direction needs to be optimized for the specific problem. This is easier to define for discrete states and finite number of neighbors (e.g. SA was successfully used to solve the traveling salesman problem which is an NP problem).

Simulated Tempering and Beyond

- Same idea for MCMC: useful if we have many different peaks of posterior distribution (multi-modal)
- We have K+1 sampling prob. $p_k=p(x)^{1/T_k}$ with $T_0=1$.
- We allow jumps from one to another, such that an increase in p is always accepted and a decrease accepted with $r=\frac{c_j q_j(\theta^{t+1})J_{j,s^t}}{c_{s^t} q_{s^t}(\theta^{t+1})J_{s^t,j}}$ and the constants c are chosen so that each k spends about the same time. At the end we only accept k=0 events.
- Parallel tempering is similar but with K+1 actual chains and flipping between them. In language of genetics this is recombination (vs mutation for SA/ST).
- Non-convex optimization and multi-modal posteriors are hard problems and there are many attempts in the literature that attempt to solve them, most of them stochastic: quantum annealing, sequential MC, stochastic tunneling, tabu search, graduated optimization (smoothing loss function), particle swarm and ant colony optimizations…

Hamiltonian Monte Carlo (HMC)

- Borrows heavily from physics: the basic idea is to add a momentum to the potential (log-posterior), and then simulate the dynamics of this system using Hamiltonian equations.
- We have seen this idea before: adding momentum to gradient descent improves the convergence towards the minimum.
- Since Hamiltonian is conserved acceptance rate is ”almost” 1.
- Here we are sampling so we need to ensure that we converge to the target distribution: the system is evolved for a few integration steps and then momentum is randomized again: this guarantees that after the marginalization over the momentum we get the right posterior.
- It requires gradient of log posterior w.r.t. parameters: this is also shared with optimization codes. It is sometimes easy, but sometimes not. Automatic differentiation methods help.

HMC Continued.

- Let us consider parameter vector x and write –log posterior $P(x)$ as $E(x)$

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z}$$

- Z is Bayes factor (evidence)
- To this we will add momentum term $K(p)$ to get full Hamiltonian H

$$H(\mathbf{x},\mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p}) \qquad K(\mathbf{p}) = \mathbf{p}^{\mathrm{T}}\mathbf{p}/2.$$

- Joint density is separable, so discarding p we get samples of $P(x)$

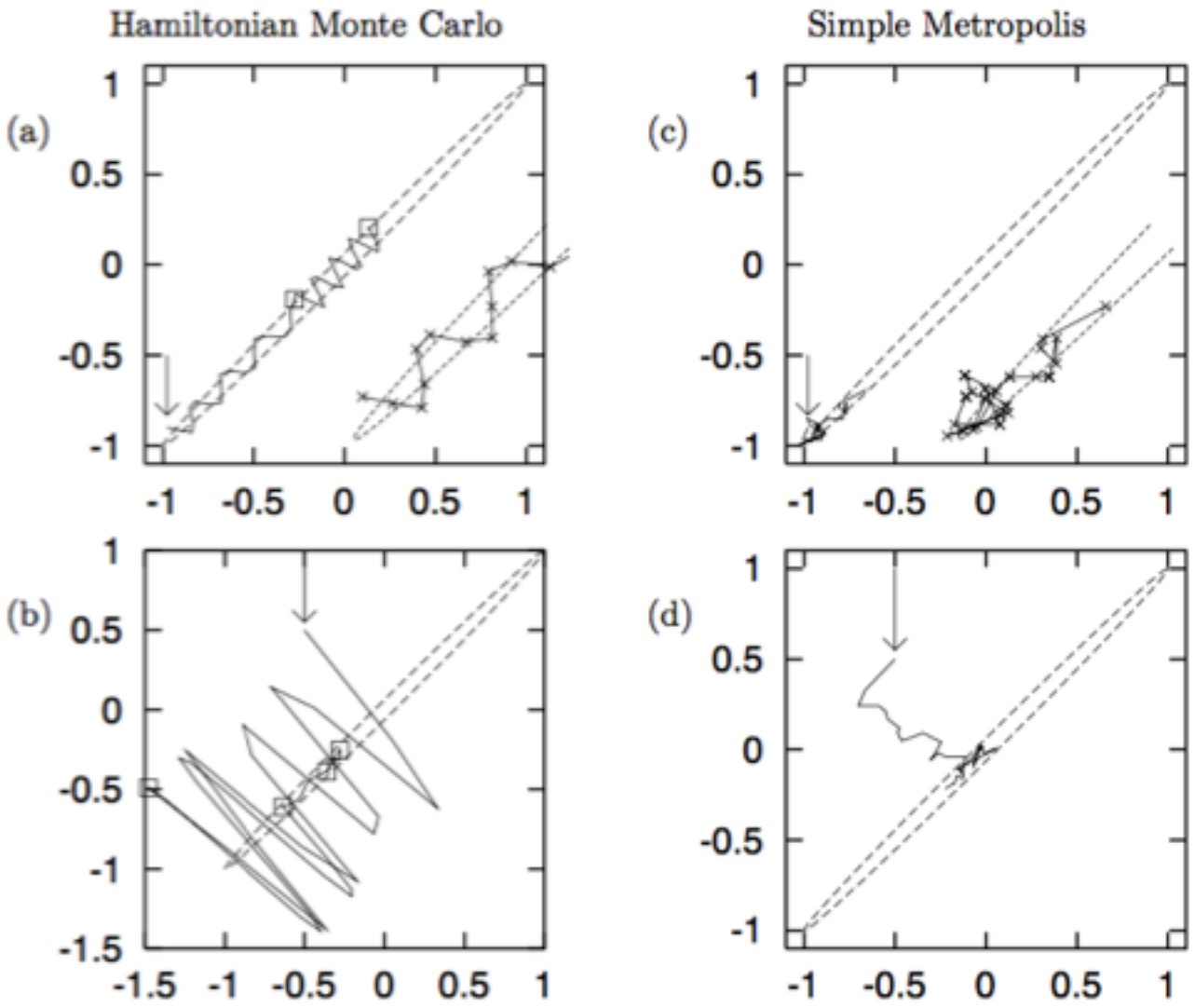
$$P_H(\mathbf{x},\mathbf{p}) = \frac{1}{Z_H}\exp[-H(\mathbf{x},\mathbf{p})] = \frac{1}{Z_H}\exp[-E(\mathbf{x})]\exp[-K(\mathbf{p})]$$

- We sample p from $\exp[-K(\mathbf{p})]/Z_K$. This is always accepted (Gibbs sampling update).

HMC Continued.

- Next we evolve the system under equations of motion $\dot{\mathbf{x}} = \mathbf{p}$, $\dot{\mathbf{p}} = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
- This is accepted with Metropolis rule, but since H is conserved rejections occur only because of numerical inaccuracy.
- Advantage is that the motion of *x* is in the direction of momentum, so the state moves a distance that is linear in computer time; typically 10-20 steps before momentum is randomized again. This reduces correlation of samples.
- One needs to discretize the dynamical equations. One popular approach is to use leap-frog integrator which is symplectic (preserves phase space). This will be discussed further when we study ordinary diff. eq.
- One can also introduce mass m, which relates momentum p to velocity, $dx/dt=p/m$. Note that this is a vector, so there is some tuning.
- We can generalize it to a mass matrix $K=pM^{-1}p/2$

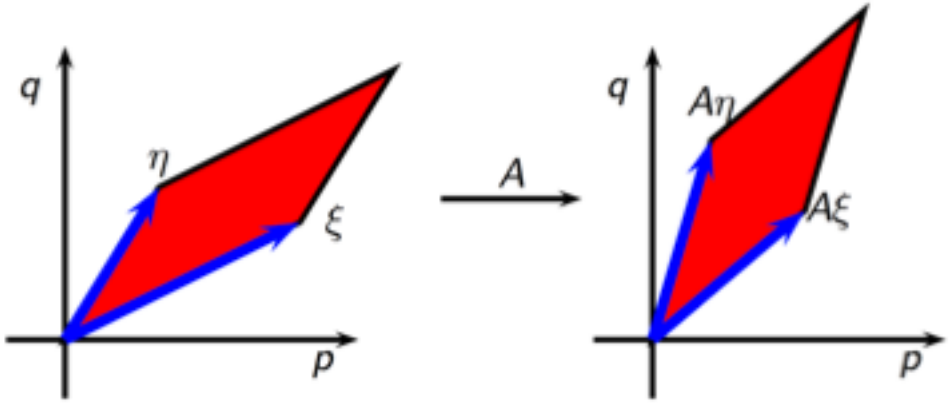
- 2 sequences of 19 steps
- 4 sequences over average 19 steps



Back to Lecture 1: Symplectic Integrators

Hamiltonian problem $\dot{p} = -H_q(p,q),\ \dot{q} = H_p(p,q)$

- Symplectic integrators preserve phase space (p,q) volume: p,q must be canonical variables
- Symplectic transformation preserves phase space area (p,q) (Liouville's theorem)
- Leapfrog is symplectic



- Hamiltonian is not conserved, but a related quantity is and one does not accumulate amplitude error, only phase error
- Useful if one needs to integrate a system for a long time: sampling!

Hamiltonian Monte Carlo

- We take a few steps integrating Hamiltonian along the path, then resample the momentum. Acceptance rate is 1 if Hamiltonian is conserved, otherwise it drops. So we'd like to move as far as possible to reduce the correlation of samples, while preserving the Hamiltonian.
- In hierarchical Bayesian models we work with many latent variables that we marginalize over: it is important that HMC solver conserves H in very high dimensions
- Symplectic integrators are ideal for this purpose. Typically we use leap-frog (2^{nd} order, symplectic)
- Codes like Stan are currently best on the market
- However, we need gradient: sometimes hard to get

Langevin style Monte Carlo

- Another way to introduce noise is to add it to the optimization equations (Welling and The 2011, Ma etal 2015)
- We have seen before that stochastic gradient descent adds noise, but we can also add it explicitly
- This leads to Langevin equation, which can be analyzed using Fokker-Planck equation (Brownian motion, random walk)

$$\mathrm{d}\mathbf{z} = \mathbf{f}(\mathbf{z})\mathrm{d}t + \sqrt{2\mathbf{D}(\mathbf{z})}\mathrm{d}\mathbf{W}(t)$$

$$\mathbf{f}(\mathbf{z}) = -\left[\mathbf{D}(\mathbf{z}) + \mathbf{Q}(\mathbf{z})\right]\nabla H(\mathbf{z}) + \Gamma(\mathbf{z}), \quad \Gamma_i(\mathbf{z}) = \sum_{j=1}^{d} \frac{\partial}{\partial \mathbf{z}_j}\left(\mathbf{D}_{ij}(\mathbf{z}) + \mathbf{Q}_{ij}(\mathbf{z})\right)$$

- For any symmetric D and antisymmetric Q the process converges to a stationary distribution p(z)=exp(-H(z))
- Here z=(q,p) and H=E+K
- dW(t) is a Wiener process, ie adds random noise component
- D or Q can be related to inverse Hessian: Q is like mass matrix

Summary

- Stochastic methods in Bayesian statistics started in 30's (Fermi), major developments in 50's (MC and MCMC), 80's (HMC, SA) and later.
- This is an active area that has been fueled by physics and related fields, specially connections to statistical physics and classical mechanics.
- MC integrals also of huge relevance for high dim
- If you have gradients then HMC is most powerful (use PyMC3, Stan), if you can write conjugate marginal use Gibbs (BUGS/JAGS, PyMC2), otherwise use Metropolis MCMC or similar alternatives (PMC, emcee…)
- For multimodal problems or evidence use simulated annealing/tempering or other specialized software (MultiNest, DNest4…)

Literature

- *Numerical Recipes*, Press et al., Chapter 7
- Computational Physics, M. Newman, Chapter 10
- *Bayesian Data Analysis,* Gelman et al. , Chapter 10-12
- D. Mackay, *Information Theory, Inference, and Learning Algorithms* (See course website), Chapter 29-30
- https://arxiv.org/pdf/1706.01520.pdf
- http://nbviewer.jupyter.org/url/astro.uchicago.edu/%7Eandrey/classes/150404_mayacamas/mcmc_convergence.ipynb
- https://github.com/KIPAC/StatisticalMethods/blob/master/chunks/montecarlo1.ipynb