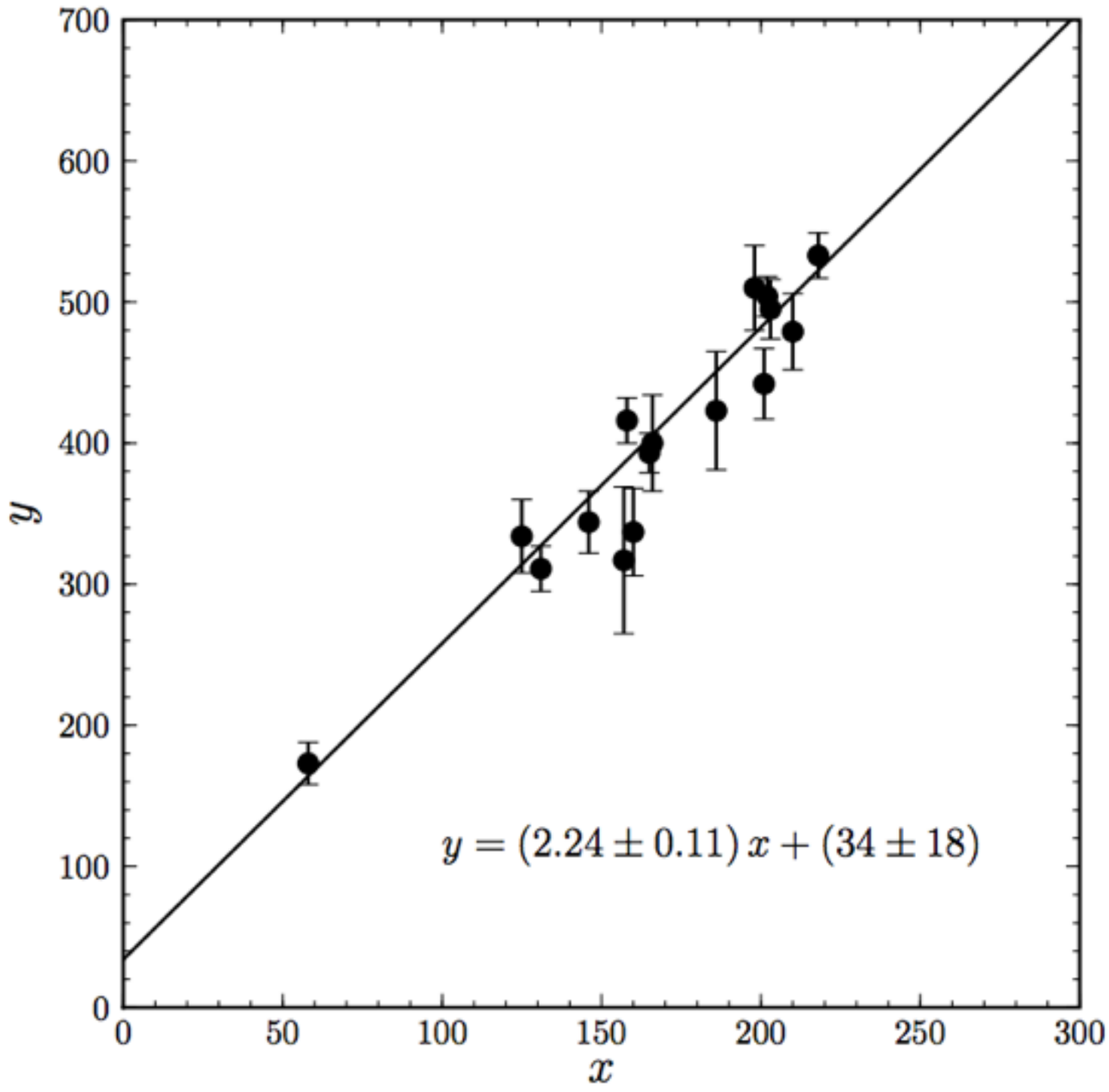
Statistics and Inference in Astrophysics

Zhu & Menard (2013)

Today: methods for assessing uncertainty in model fits

- Bayesian: sampling the posterior probability distribution, in particular, Markov Chain Monte Carlo methods

- Frequentist: non-parametric methods: bootstrap, jackknife



Hogg, Bovy, & Lang (2010)

Fitting a line

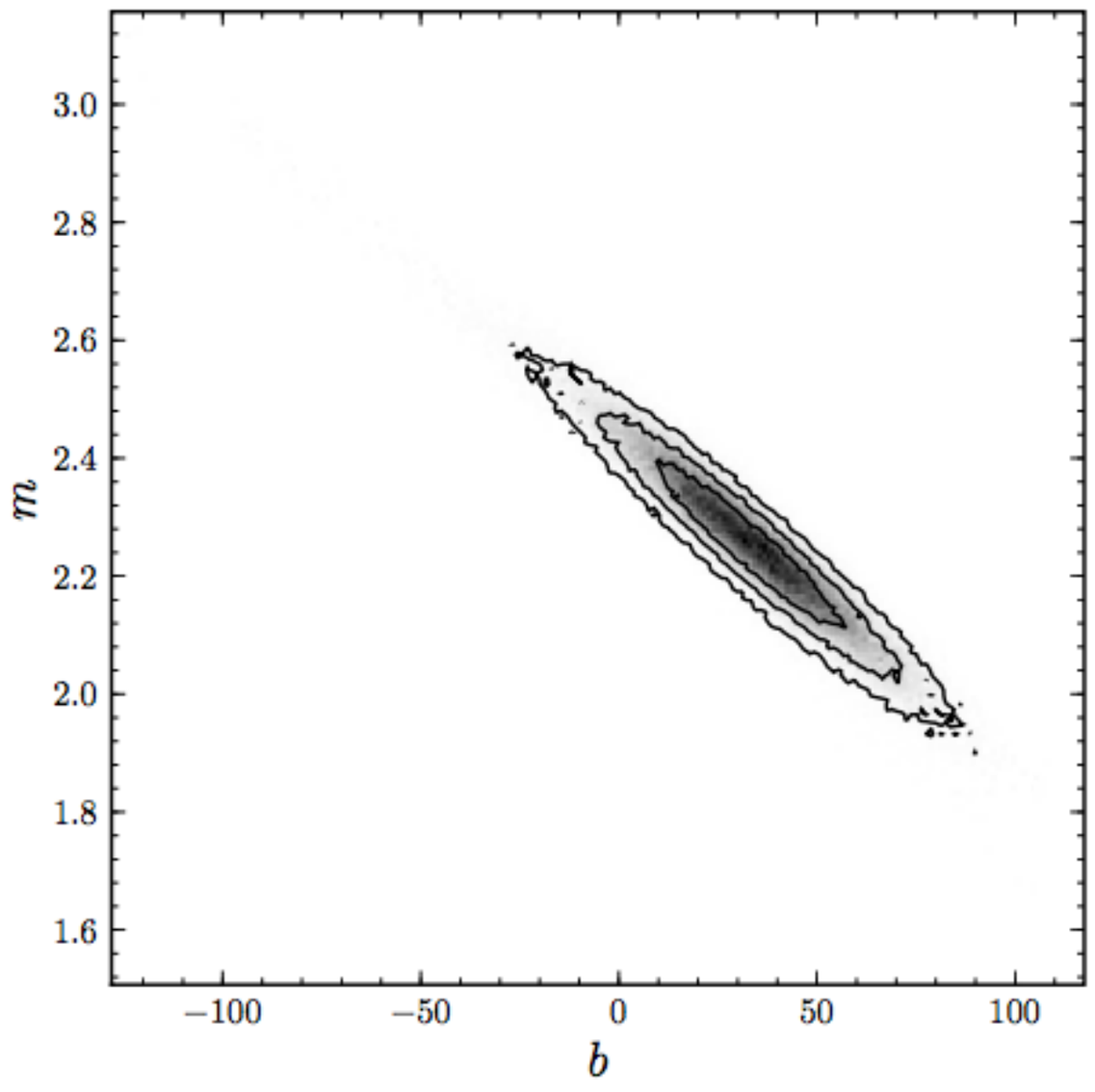
- Straight line model has two parameters: slope *m* and intercept *b*

- Likelihood, single point: $p(y_i|m,b,x_i,\sigma_{y,i}) = N(y_i|mx_i+b,\sigma^2_{y,i})$

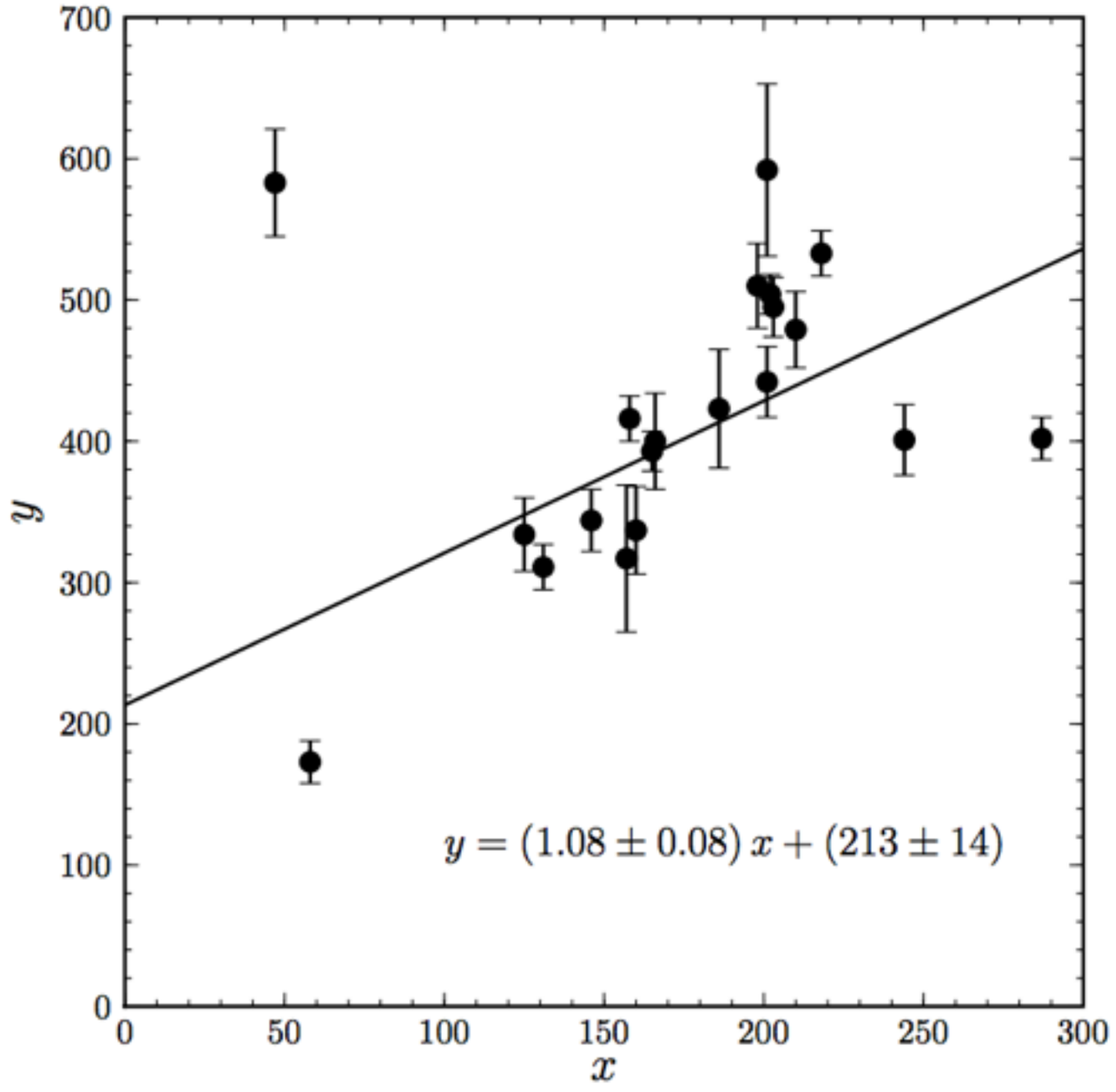
- Independent data points:
  $p(\{y\}|m,b,\{x\},\{\sigma_y\}) =$
  $p(y_0|m,b,x_0,\sigma_{y,0})$ x $p(y_1|m,b,x_1,\sigma_{y,1})$x… x $p(y_{N-1}|m,b,x_{N-1},\sigma_{y,N-1})$

- Posterior:
  $p(m,b|\{y\},m,b,\{x\},\{\sigma_y\} \sim p(\{y\}|m,b,\{x\},\{\sigma_y\})$ x $p(m,b)$

- Two parameters, so easy to optimize, grid-evaluate,…


3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
m
−100
−50
0
50
100
b

Mixture model for outliers



Mixture model for outliers

- Model outliers using a *mixture model*: each data point has some probability q_i to be actually drawn from the line, and probability $(1-q_i)$ to be drawn from a background model $p_{bg}(y_i|x_i,\sigma_{y,i},\ldots)$

- Simple background model:
  $p_{bg}(y_i|x_i,\sigma_{y,i},\ldots) = N(y|Y_b,V_b+\sigma^2_{y,i})$

Mixture model for outliers

In this case, the likelihood is

$$
\begin{aligned}
\mathscr{L} &\equiv p(\{y_i\}_{i=1}^N | m, b, \{q_i\}_{i=1}^N, Y_\mathrm{b}, V_\mathrm{b}, I) \\
\mathscr{L} &= \prod_{i=1}^N \left[p_\mathrm{fg}(\{y_i\}_{i=1}^N | m, b, I)\right]^{q_i} \left[p_\mathrm{bg}(\{y_i\}_{i=1}^N | Y_\mathrm{b}, V_\mathrm{b}, I)\right]^{[1-q_i]} \\
\mathscr{L} &= \prod_{i=1}^N \left[\frac{1}{\sqrt{2\,\pi\,\sigma_{yi}^2}} \exp\left(-\frac{[y_i - m\,x_i - b]^2}{2\,\sigma_{yi}^2}\right)\right]^{q_i} \\
&\quad \times \left[\frac{1}{\sqrt{2\,\pi\,[V_\mathrm{b} + \sigma_{yi}^2]}} \exp\left(-\frac{[y_i - Y_\mathrm{b}]^2}{2\,[V_\mathrm{b} + \sigma_{yi}^2]}\right)\right]^{[1-q_i]} ,
\end{aligned}
$$

Posterior requires prior on q_i, introduces new parameter P_b

$$
\begin{aligned}
p(m, b, \{q_i\}_{i=1}^N, P_\mathrm{b}, Y_\mathrm{b}, V_\mathrm{b} | I) &= p(\{q_i\}_{i=1}^N | P_\mathrm{b}, I)\, p(m, b, P_\mathrm{b}, Y_\mathrm{b}, V_\mathrm{b} | I) \\
p(\{q_i\}_{i=1}^N | P_\mathrm{b}, I) &= \prod_{i=1}^N [1 - P_\mathrm{b}]^{q_i}\, P_\mathrm{b}^{[1-q_i]} ,
\end{aligned}
$$

Hogg, Bovy, & Lang (2010)

Mixture model for outliers

- Parameters of the model are now: *m*, *b*, Y_b, V_b, P_b, q_0, q_1, …, q_{N-1} —> N+5 parameters!

- Efficiently exploring the posterior PDF becomes much harder; grid-evaluation impossible!

- Note: we can analytically marginalize over q_i

$$\mathscr{L} \equiv \prod_{i=1}^{N} \left[(1 - P_\mathrm{b})\, p_\mathrm{fg}(\{y_i\}_{i=1}^{N} | m, b, I)) + P_\mathrm{b}\, p_\mathrm{bg}(\{y_i\}_{i=1}^{N} | Y_\mathrm{b}, V_\mathrm{b}, I) \right]$$

$$\mathscr{L} \propto \prod_{i=1}^{N} \left[\frac{1 - P_\mathrm{b}}{\sqrt{2\,\pi\,\sigma_{yi}^2}} \exp\left(-\frac{[y_i - m\,x_i - b]^2}{2\,\sigma_{yi}^2} \right) + \frac{P_\mathrm{b}}{\sqrt{2\,\pi\,[V_\mathrm{b} + \sigma_{yi}^2]}} \exp\left(-\frac{[y_i - Y_\mathrm{b}]^2}{2\,[V_\mathrm{b} + \sigma_{yi}^2]} \right) \right] ,$$

Sampling methods for the posterior PDF

- Most things that want to do with the PDF $p(\theta)$ involve integrals over the PDF:
  - Mean = $\int d\theta\, p(\theta)\, \theta$
  - Median: $\int^{\text{median}} d\theta\, p(\theta) = \int_{\text{median}} d\theta\, p(\theta)$
  - Variance = $\int d\theta\, p(\theta)\, \theta^2 - [\int d\theta\, p(\theta)\, \theta]^2$
  - Quantiles: $\int^{\text{quantile } \theta} d\theta\, p(\theta) = \text{quantile} \times \int d\theta\, p(\theta) = \text{quantile}$
  - Marginalization: $p(\theta) = \int d\eta\, p(\theta,\eta)$
- None of these care about the overall normalization of $p(\theta)$ [set $\int d\theta\, p(\theta) = 1$]
- Therefore, can use Monte Carlo integration techniques

Monte Carlo Integration

- Multi-dimensional integral

$$\int \mathrm{d}\theta \, f(\theta) = V \times \frac{1}{N} \sum_i f(\theta_i)$$

where

$$V = \int \mathrm{d}\theta$$

θ_i are uniformly sampled points within the domain of θ

Monte Carlo Integration

- No need to use uniform sampling, can just as easily do

$$\int \mathrm{d}\theta\, f(\theta) = \int \mathrm{d}\theta\, q(\theta)\, \frac{f(\theta)}{q(\theta)}$$

$$= V_q \times \frac{1}{N} \sum_i \frac{f(\theta_i)}{q(\theta_i)}$$

where

$$V_q = \int \mathrm{d}\theta\, q(\theta)$$

θ_i are points sampled from $q(\theta)$

- If you choose $q(\theta)$ that closely follows $f(\theta)$, $f(\theta_i)/q(\theta_i) \sim 1$ and integral will quickly converge

Monte Carlo Integration for probability distributions

- Back to our integrals of the form $\int d\theta\, p(\theta)\, f(\theta)$

- Using Monte-Carlo integration

$$\int \mathrm{d}\theta\, p(\theta)\, f(\theta) = \frac{1}{N} \sum_i f(\theta_i)$$

  if θ_i sampled from $p(\theta)$, because $V_p = \int d\theta\, p(\theta) = 1$

- So all integrals of the posterior PDF can be performed using Monte Carlo integration, if we can efficiently sample $p(\theta)$!

Importance sampling

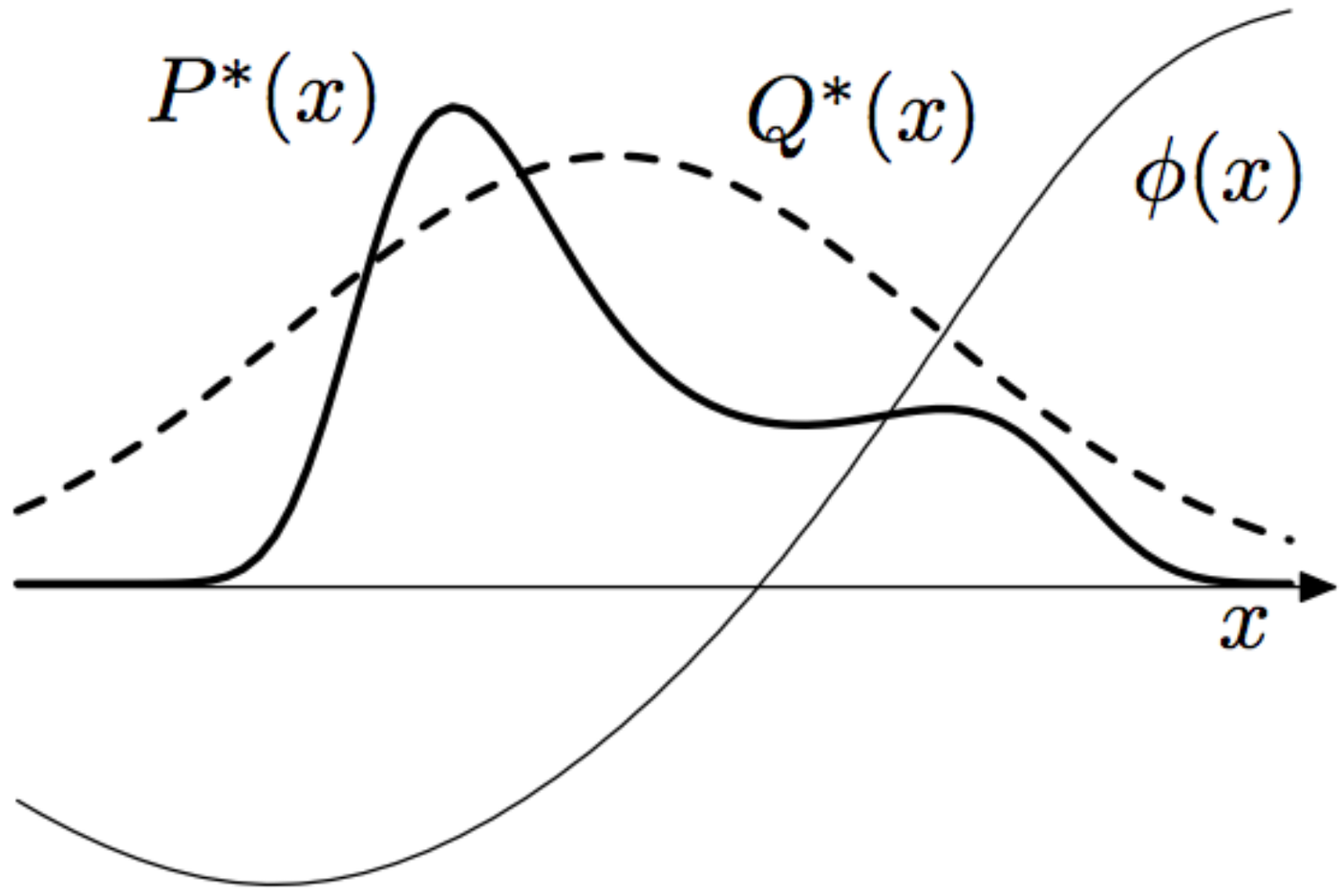
- Sampling p(θ) is hard! So let's sample a *different* distribution that is easy to sample q(θ) and use

$$\int \mathrm{d}\theta\, p(\theta)\, f(\theta) = \int \mathrm{d}\theta\, q(\theta)\, \frac{p(\theta)}{q(\theta)}\, f(\theta)$$

$$= \frac{1}{N} \sum_i \frac{p(\theta_i)}{q(\theta_i)}\, f(\theta_i)$$

- The $p(\theta_i)/q(\theta_i)$ are known as the *importance* weights
- They re-weight the importance of each sample
- Works well if q(θ) is close to p(θ), otherwise introduces large variance: think about what happens when q(θ) is small when p(θ) is large!

Importance sampling



Mackay (2003)

Importance sampling

- Useful in some contexts:

  For example, somebody gave you samples from a posterior PDF with a prior that you don't like —>

  You want $\int \mathrm{d}\theta\, p_{\mathrm{you}}(\theta|\mathrm{data})\, f(\theta) \propto \int \mathrm{d}\theta\, p(\mathrm{data}|\theta)\, p_{\mathrm{you}}(\theta)\, f(\theta)$

- But you have samples θ_i from

  $$p_{\mathrm{not\ you}}(\theta|\mathrm{data}) \propto p(\mathrm{data}|\theta)\, p_{\mathrm{not\ you}}(\theta)$$

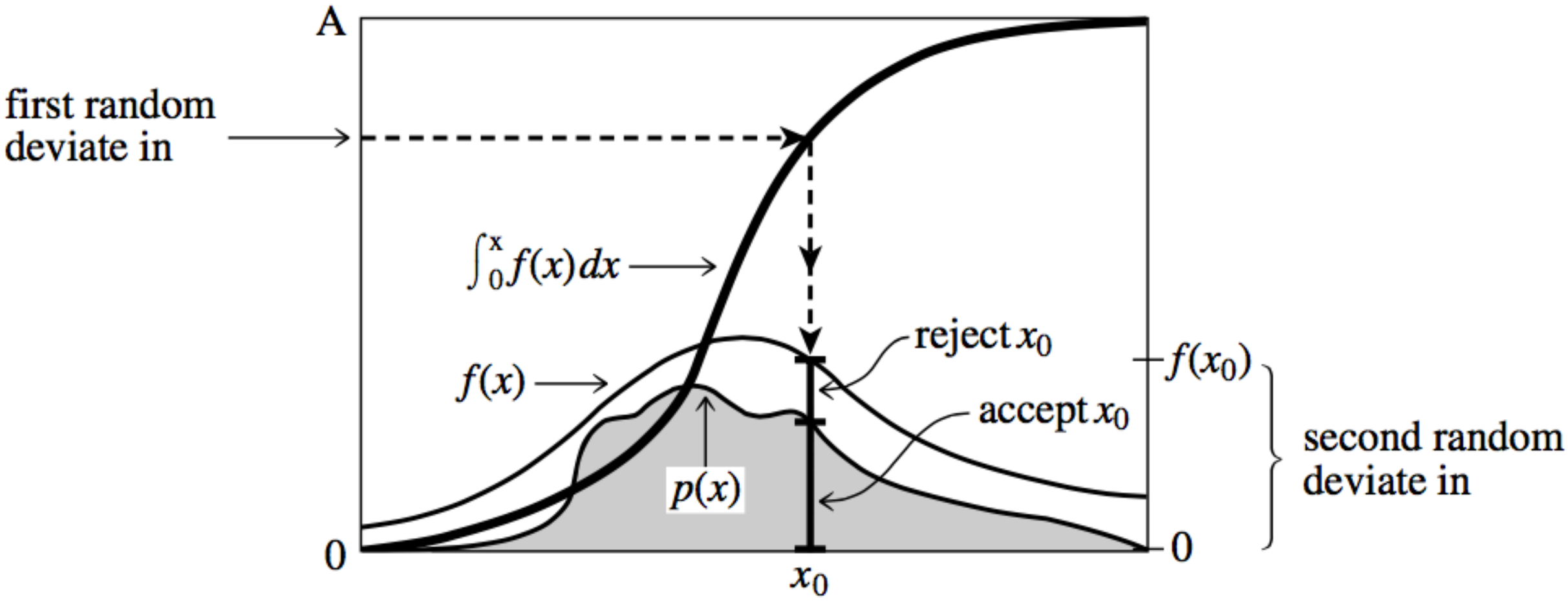
- Can do

  $$\int \mathrm{d}\theta\, p_{\mathrm{you}}(\theta|\mathrm{data})\, f(\theta) = \frac{1}{N} \sum_i \frac{p_{\mathrm{you}}(\theta)}{p_{\mathrm{not\ you}}(\theta)}\, f(\theta)$$

  which should be fine as long as the prior doesn't change too much

Rejection sampling

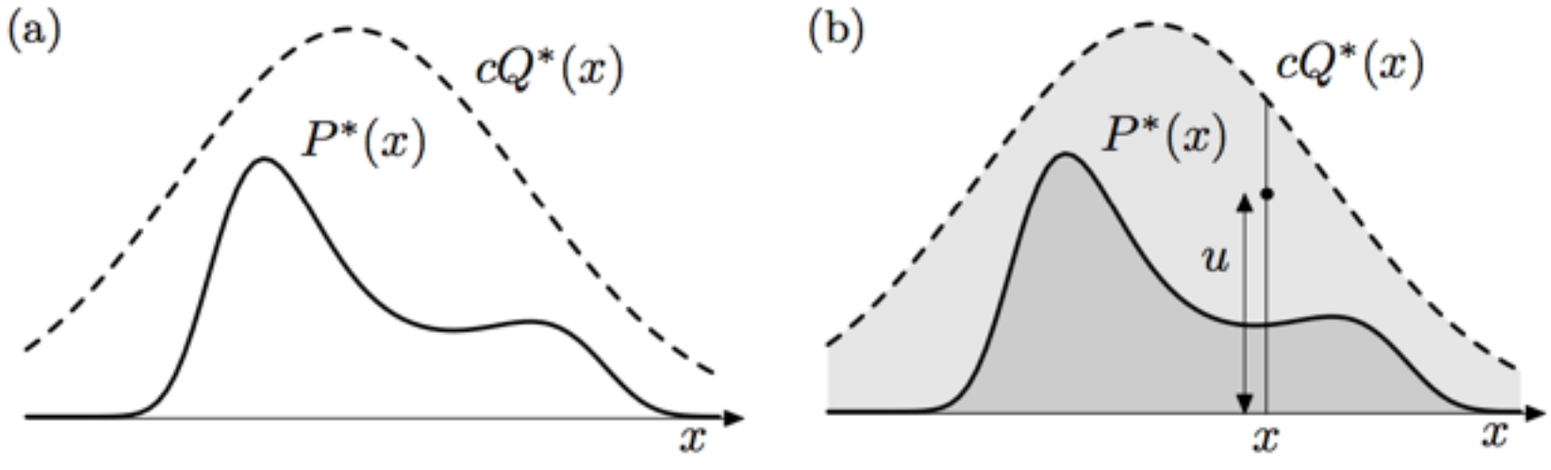
- Sampling from p(b) == uniformly sampling area under p(b)



Numerical recipes (2007)

Rejection sampling

- Have q(x) such that c x q(x) always > p(x)
- q(x) easy to sample (e.g., uniform or Gaussian)
- Sample *v* from q(x) and *u* from Uniform(0,1)
  if *u* < p(v)/q(*v*)/c: return *v*
  else: try again



Mackay (2003)

Rejection sampling

- Works well in 1D, but difficult for multi-dimensional distributions, because volume under q(x) and that under p(x) quickly becomes very different

- Even in 1D it can be difficult to find a q(x)

- Importance sampling and rejection sampling useful *because each sample is independent*

Markov chains

- A Markov chain is a chain of randomly produced samples (*states*) for which the transition probability to the next state only depends on the current state, not the previous history —> *memoryless*

- Markov chain defined by transition probability T(x';x) which gives the probability of going to x' when you're currently at x

- Markov Chain Monte Carlo methods construct T(x';x) such that the chain samples a given p(x)

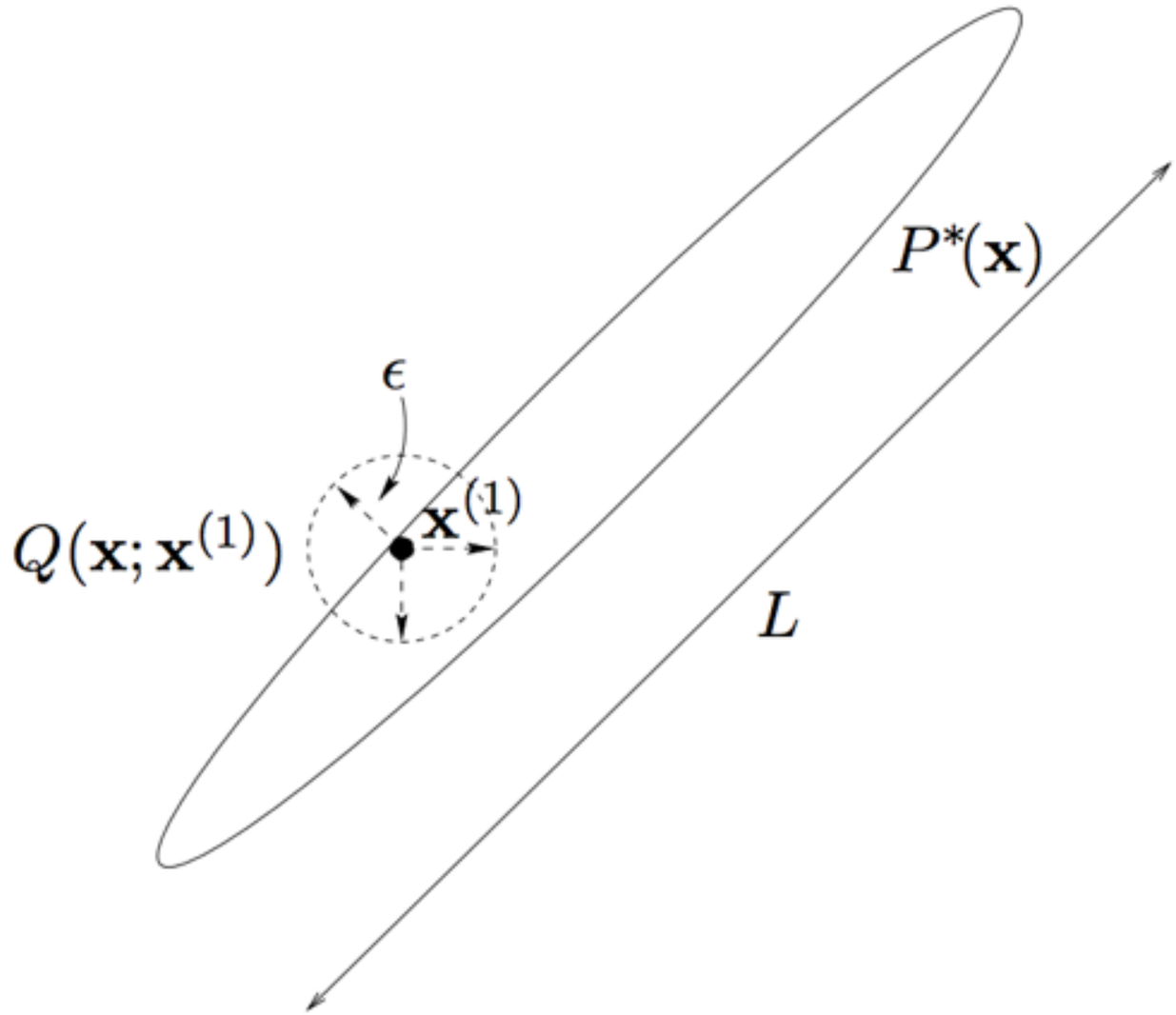
Metropolis-Hastings

- Want to sample p(x)

- Proposal distribution q(x';x) [this is *not* the T(x';x) from previous slide!]; For example, Gaussian centered on x with some width

- Algorithm: you're at x_i
    1. Draw from x_t from $q(x_t;x_i)$
    2. Compute $a = [p(x_t)\ q(x_i;x_t)] / [p(x_i)\ q(x_t;x_i)]$
    3. If $a > 1$: accept x_t; else: accept x_t with probability a
    4. If accepted: $x_{i+1} = x_t$; else: $x_{i+1} = x_i$ —> always add!

Metropolis-Hastings



Mackay (2003)

Metropolis-Hastings: special case of a symmetric proposal distribution

- Algorithm: you're at x_i
    1. Draw from x_t from $q(x_t;x_i)$
    2. Compute $a = [p(x_t)\ q(x_i;x_t)] / [p(x_i)\ q(x_t;x_i)]$
       $\mathbf{= p(x_t) / p(x_i)}$
    3. If $a > 1$: accept x_t; else: accept x_t with probability a
    4. If accepted: $x_{i+1} = x_t$; else: $x_{i+1} = x_i$

- So, if proposed state has higher probability, *always accept*

- But can go to lower probability region with some probability —> *not an optimizer!*

Metropolis-Hastings in practice

- Need to choose q(x';x) —> often a Gaussian centered on x, with some width, in higher dimensions typically spherical Gaussian

- Width is adjustable parameter: should be *O*(width of p[x])

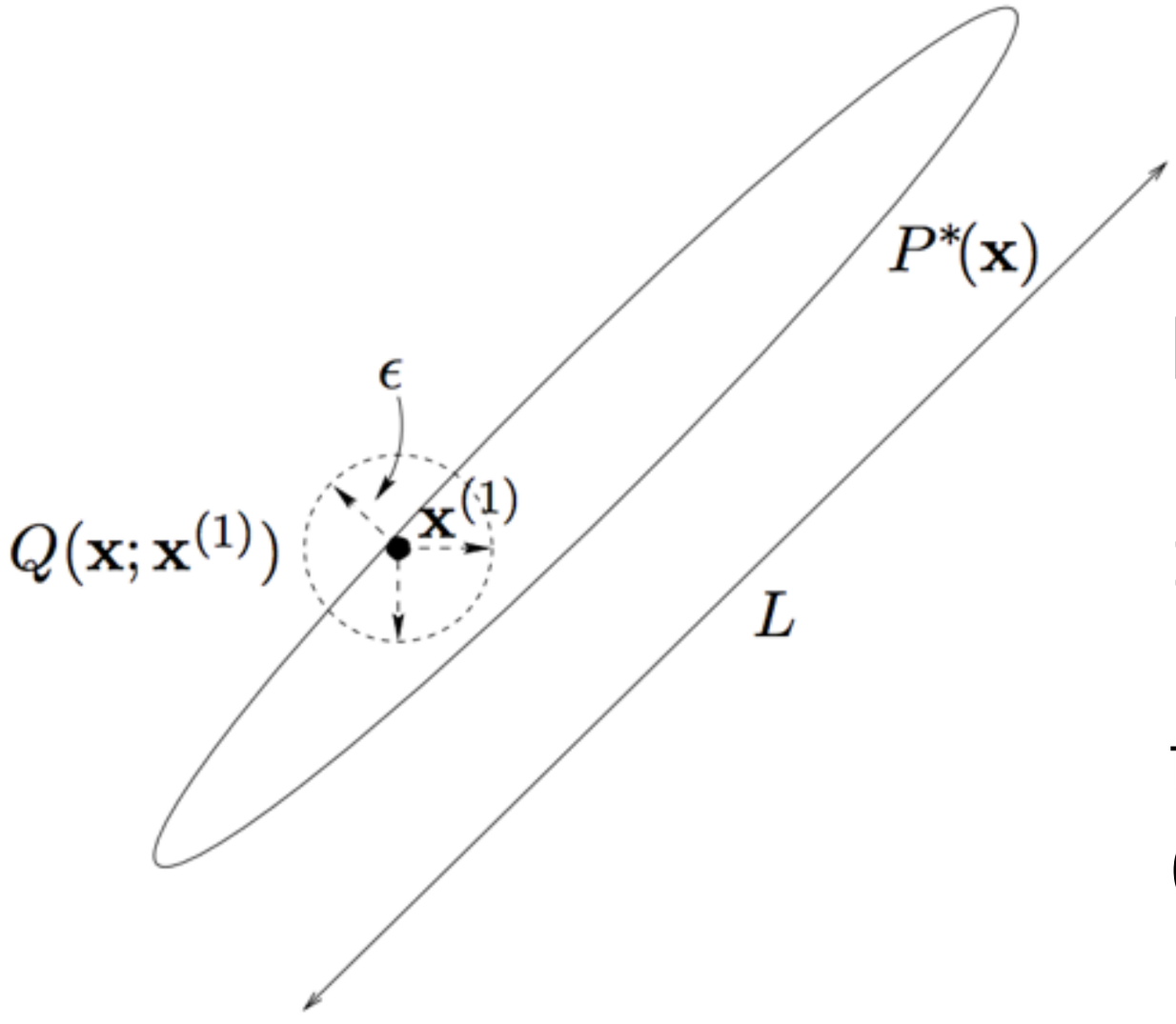
  Set it too large: jump to regions with low p(x) —> reject

  Set it too small: jump to regions with very similar p(x) —> Transition probability ~1 —> accept most, but don't explore

- Typically needs a lot of adjusting; acceptance fraction = (# of times x_t =/= x_i) / (total # of steps)

- Theoretical work has shown that optimal acceptance fraction in 1D = 50%, in higher-D 23% (Roberts & Gelman 1997)

Metropolis-Hastings



Need on order of

>$(L/width)^2$ steps

to explore the PDF (random walk)

Mackay (2003)

Markov Chain Monte Carlo generalities

- When and why do MCMC algorithms work? Important to understand to not get tripped up in practice!

- Markov Chain characterized by transition probability T(x';x) [for MH, this is the algorithm given]

- Probability distribution $q^{i+1}(x')$ of value x' starting from probability distribution for $q^{i}(x)$:

  $q^{i+1}(x') = \int dx\ T(x';x)\ q^{i}(x)$

- So T(x';x) transforms one probability distribution into another

MCMC generalities

- For a Markov Chain algorithm to explore the desired distribution p(x) two requirements:

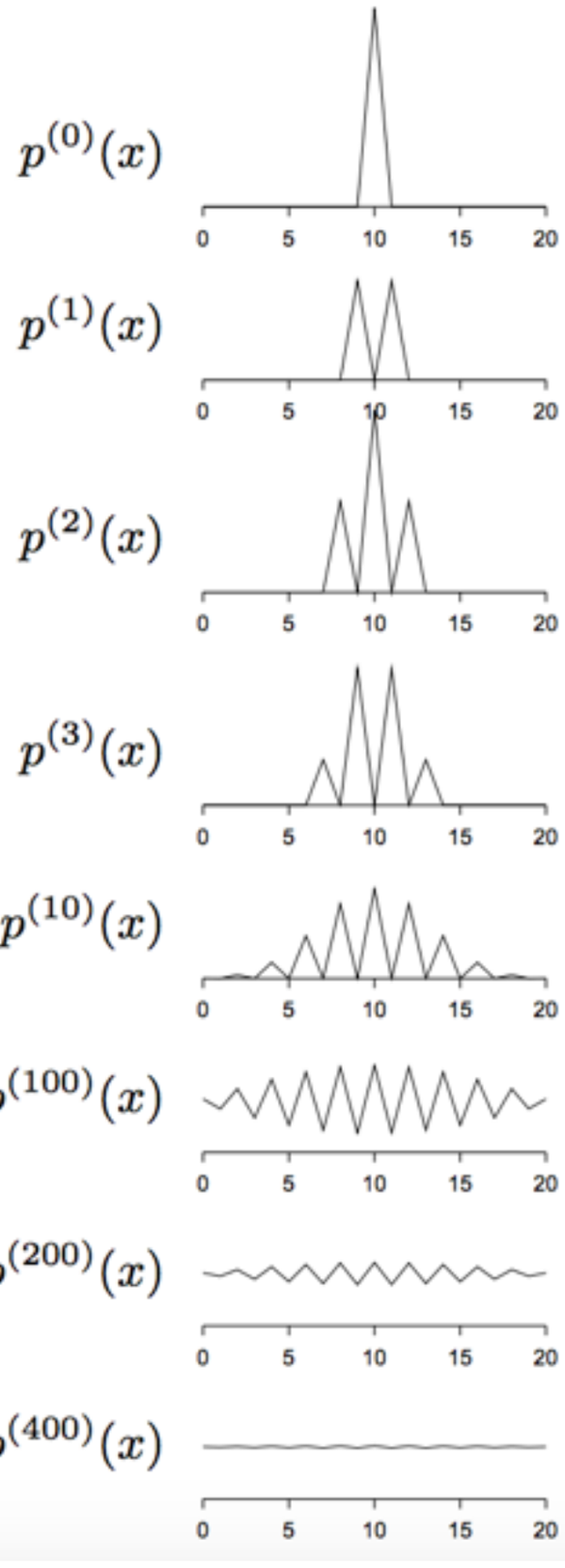
- p(x) should be an invariant distribution of the Markov Chain:

  $p(x') = \int dx\, T(x';x)\, p(x)$

- Chain must be ergodic: $q^{i+1}(x) \longrightarrow p(x)$ for $i \longrightarrow \infty$ (chain shouldn't be periodic, …)

Example: sampling a uniform distribution

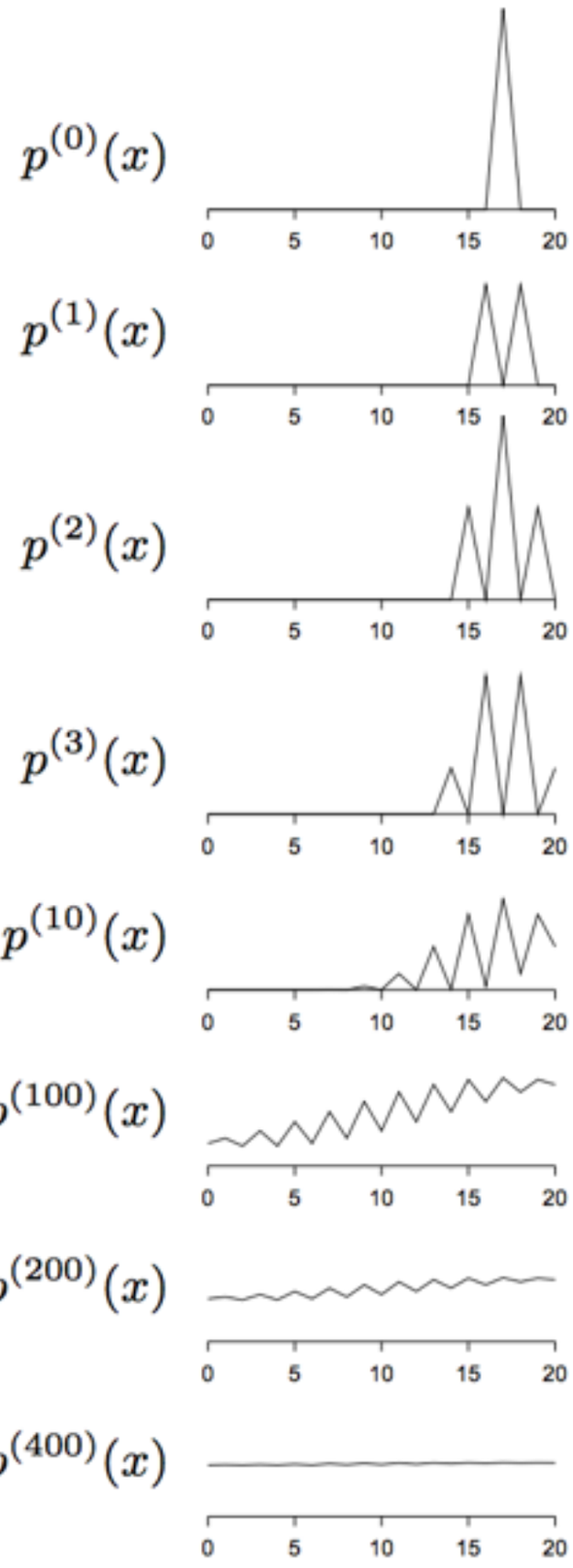
$$\mathbf{T} = \begin{bmatrix} 1/2 & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & 1/2 \end{bmatrix}$$



Mackay (2003)

Example: sampling a uniform distribution

$$\mathbf{T} = \begin{bmatrix} 1/2 & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & 1/2 \end{bmatrix}$$



Mackay (2003)

Detailed balance

- Invariance of distribution can be ensured by *detailed balance*:
- T(x';x)p(x) = T(x;x')p(x') for all x, x'
- Means that chain is *reversible*: just as likely to go from x—>x' as to go from x'—>x
- Invariance then satisfied because:

$$
\begin{aligned}
p(x') &= \int dx\, T(x';x)\, p(x) \\
&= \int dx\, T(x;x')\, p(x') \text{ [detailed balance]} \\
&= p(x') \int dx\, T(x;x') \\
&= p(x')
\end{aligned}
$$

- Sufficient, but not necessary

Metropolis-Hastings

- Pretty easy to show that MH satisfies detailed balance, but left as exercise

- How to ensure that the chain is ergodic? One simple way is to make sure that T(x';x) > 0 for all x' with non-zero p(x') [non-zero prior]

Gibbs sampling

- In multiple dimensions, say p(x,y)

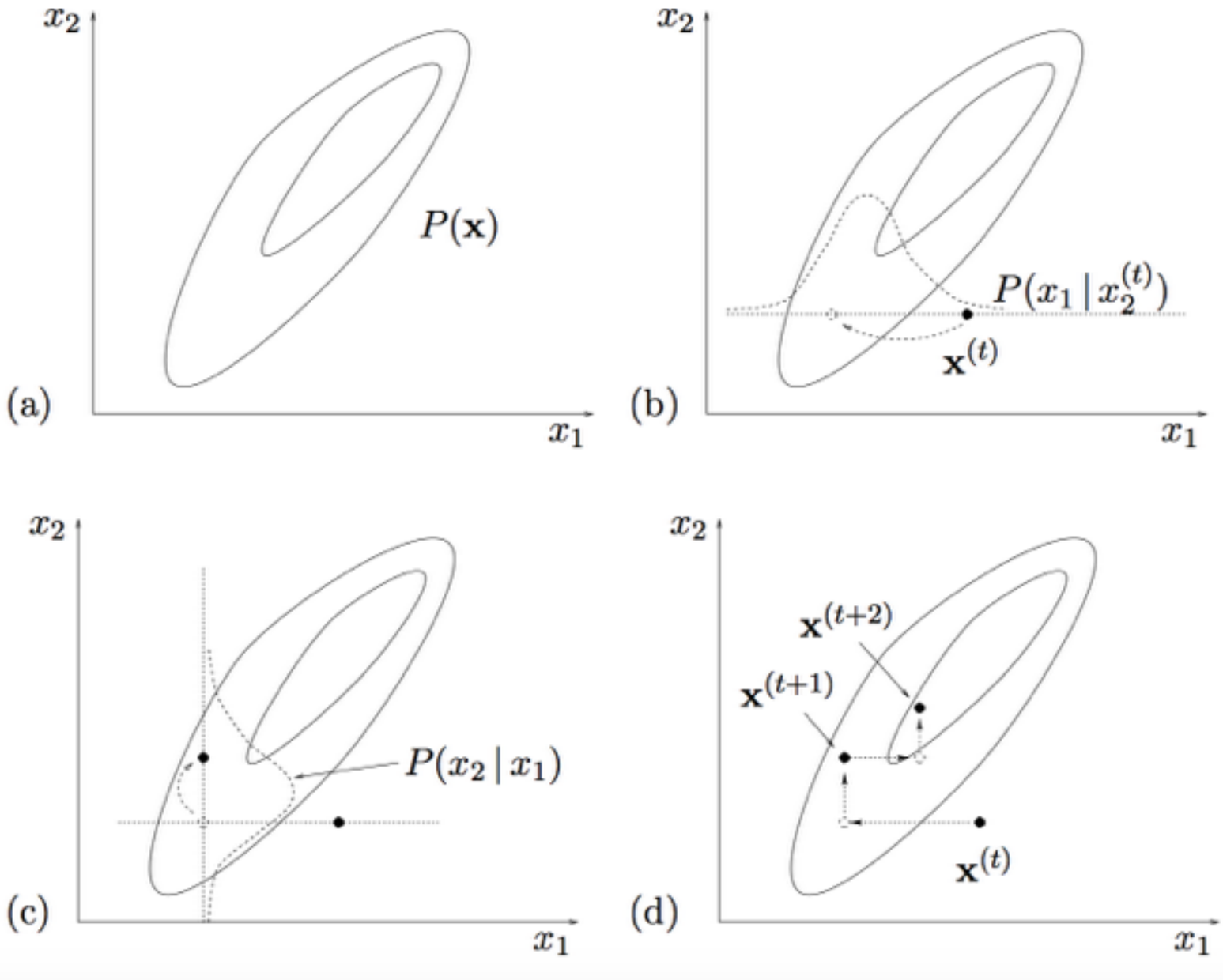
- Sample: Starting at (x_i,y_i)
    1. x_{i+1} from $p(x|y_i)$
    2. y_{i+1} from $p(y|x_{i+1})$
    3. New (x_{i+1},y_{i+1})

- Useful when:

    - Each conditional distribution is simple (or some of them)

    - Want to sample different dimensions in different ways (MH with different step sizes, more advanced sampling for some parameters)

Gibbs sampling



Mackay (2003)

Metropolis-Hastings and Gibbs sampling are nice, but typically require some adjustable step size that can lead to an unacceptable acceptance fraction

Ensemble samplers

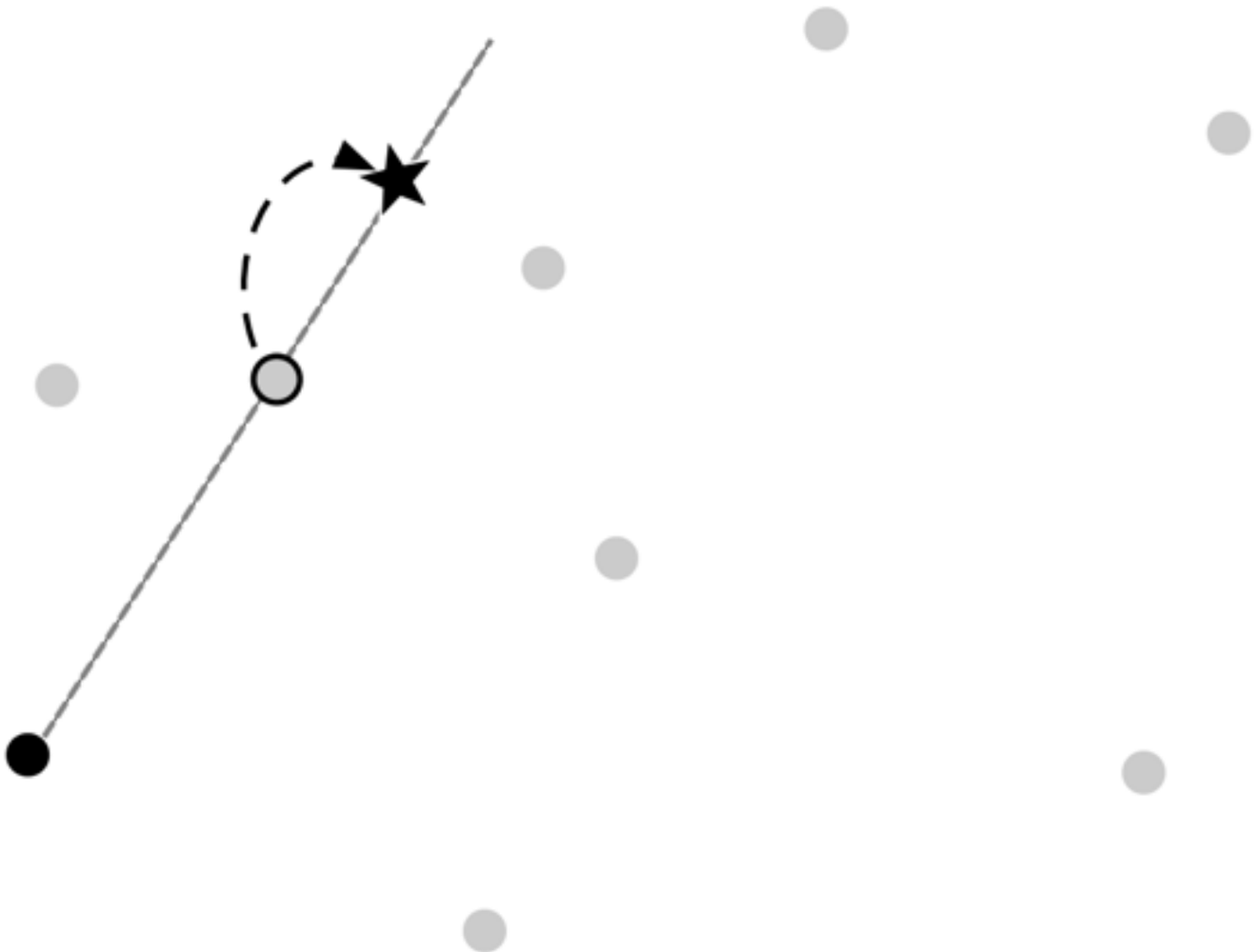
- So far have considered single sample x_i that gets updated

- Ensemble sampler have a state consisting of many samples $\{x\}_i$ that get updated by Markovian transitions

- Will focus on most popular one: affine-invariant ensemble sampler of Goodman & Weare (2009; aka, *emcee*)

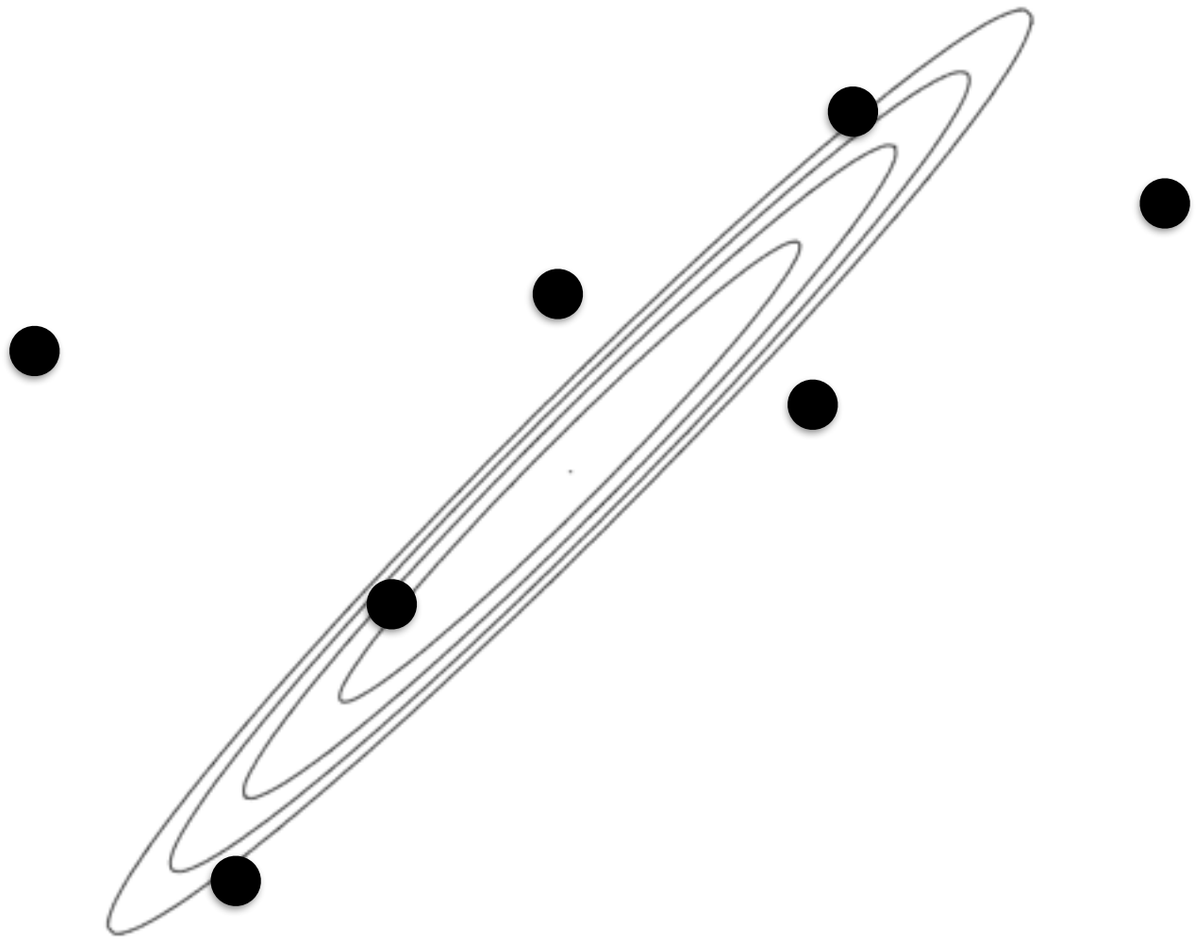
- Variations have different points in the ensemble at different temperatures, …

Affine-invariant sampler (*emcee*)



Goodman & Weare (2009)

Affine-invariant sampler (*emcee*)



Goodman & Weare (2009)

Affine-invariant sampler (*emcee*)

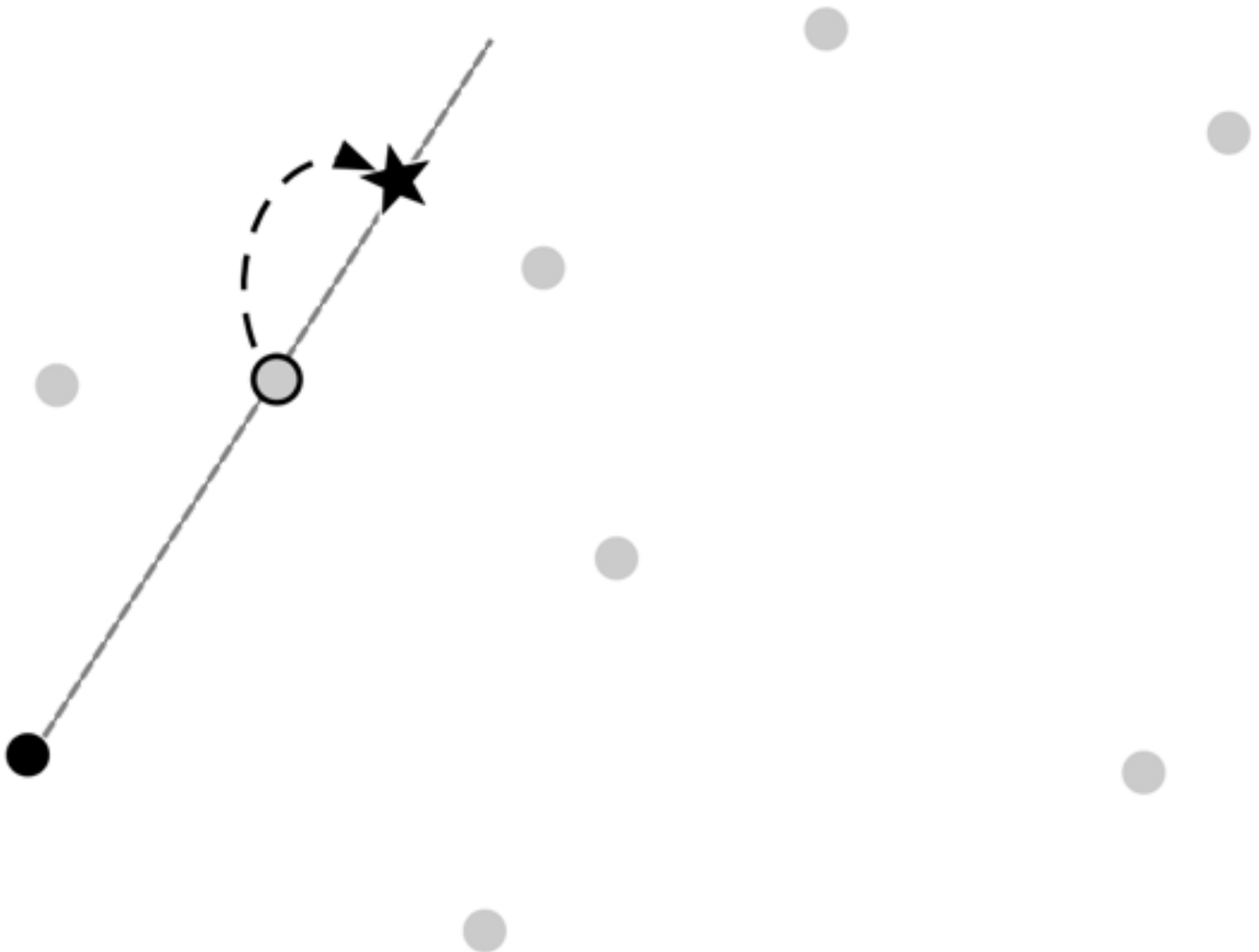
- Each x in $\{x\}_i$ is called a *walker*

- Detailed algorithm: Starting with ensemble $\{x\}_i$
    1. Loop through each walker *k:* x_k
    2. Draw a walker x_l from the set of walkers w/o *k*
    3. Draw z from g(z)
    4. Propose new $x_{k,i+1} = x_k + z(x_k - x_l)$
    5. Compute $q = z^{N-1} \times p(x_{k,i+1})/p(x_k)$
    6. Draw uniform *u* from [0,1]
    7. If q >= *u*: accept $x_{k,i+1}$; else: keep $x_{k,i}$

- 3. is called the *stretch move*; need to specify g(z)

- If g(z) satisfies g(1/z) = z g(z), the above algorithm satisfies detailed balance; g(z) = $1/\sqrt{z}$ for z in [1/a,a], a free parameter

Affine-invariant sampler (*emcee*)



Goodman & Weare (2009)

Affine-invariant sampler (*emcee*): parallel version

- Each walker needs to be updated in series in the previous algorithm —> can take a long time

- Naive parallelization (update all simultaneously using their position in iteration i) fails to satisfy detailed balance

- Can split walkers into set of two, update all walkers from one set simultaneously by only allowing moves wrt walkers in the other set —> satisfies detailed balance

Affine-invariant sampler (*emcee*)

- Algorithm needs value for a, but just scaling that can be left the same for all problems (works well)

- Need to watch out for non-ergodic chains!
    - If # of walkers < dimension of space, cannot sample entire space!
    - Should use # of walkers >> dimension of space to avoid getting stuck near subspace

- Like Metropolis-Hastings, possible that acceptance fraction is very low

emcee demo

MCMC overview

- Metropolis-Hastings: simple to implement, need to pick proposal distribution, need to monitor acceptance fraction

- Gibbs sampling: Great when (some) conditional probabilities are simple

- emcee: Insensitive to step size, so good go-to methods that don't require much supervision; good python implementation of ensemble sampler *emcee* (http://dan.iel.fm/emcee)

- All of these have *random walk behavior*: takes a long time to explore the PDF

Hamiltonian Monte Carlo

- Method to avoid random walk behavior by proposing new samples far from current point

- Does this by pretending that -ln $p(x_i)$ is a *potential energy* $U(x_i)$ and adding *N* new *momentum* variables p_i with kinetic energy $K(p_i) = \Sigma_i\, p_i^2/[2m_i]$

- Then $$p(\vec{x},\vec{p}) \propto \exp\left[-U(\vec{x}) - \sum_i \frac{p_i^2}{2m_i}\right]$$

- and $$p(\vec{x}) = \int \mathrm{d}\vec{p}\, p(\vec{x},\vec{p}) = \exp\left[-U(\vec{x})\right]$$

Hamiltonian Monte Carlo

- Now propose new points in two steps:

  Currently at (x_i,p_i):
  1. Sample p_a from $\exp(-p_i^2/[2m_i])$ —> Gaussian = simple
  2. Sample new (x_{i+1},p_{i+1}) by
     (a) simulate Hamiltonian dynamics
         (xi,pa) —> (x_p,p_p) [leapfrog]
     (b) MH accept/reject (x_p,p_p) based on ratio
         $\exp[-U(x_p)-p_p^2/\{2m\}] / \exp[-U(x_i)-p_a^2/\{2m\}]$

- If energy conserved, ratio in (b) == 1 —> *always* accept

- Step 2. is *not* a random walk and one can move to a very different point of parameter space

Hamiltonian Monte Carlo in practice

- To simulate dynamics we need the *force:* the derivative of the likelihood —> often difficult to compute by hand and numerical derivatives are unstable
- Need to set two parameters related to integration:
    - Stepsize ε
    - Number of steps L
    - Want ε to be as large as possible and still conserve energy; L such that one moves to a very different part of the PDF *but no further*

Derivatives for Hamiltonian Monte Carlo

- Derivatives are in principle easy: can use chain rule:

$$\frac{\mathrm{d}f(g(x))}{\mathrm{d}x} = f'(g[x])\,g'(x)$$

- All computer functions are in the end combinations of primitives +,-,x,/ [and at a slightly higher level exp, sin, cos, …]

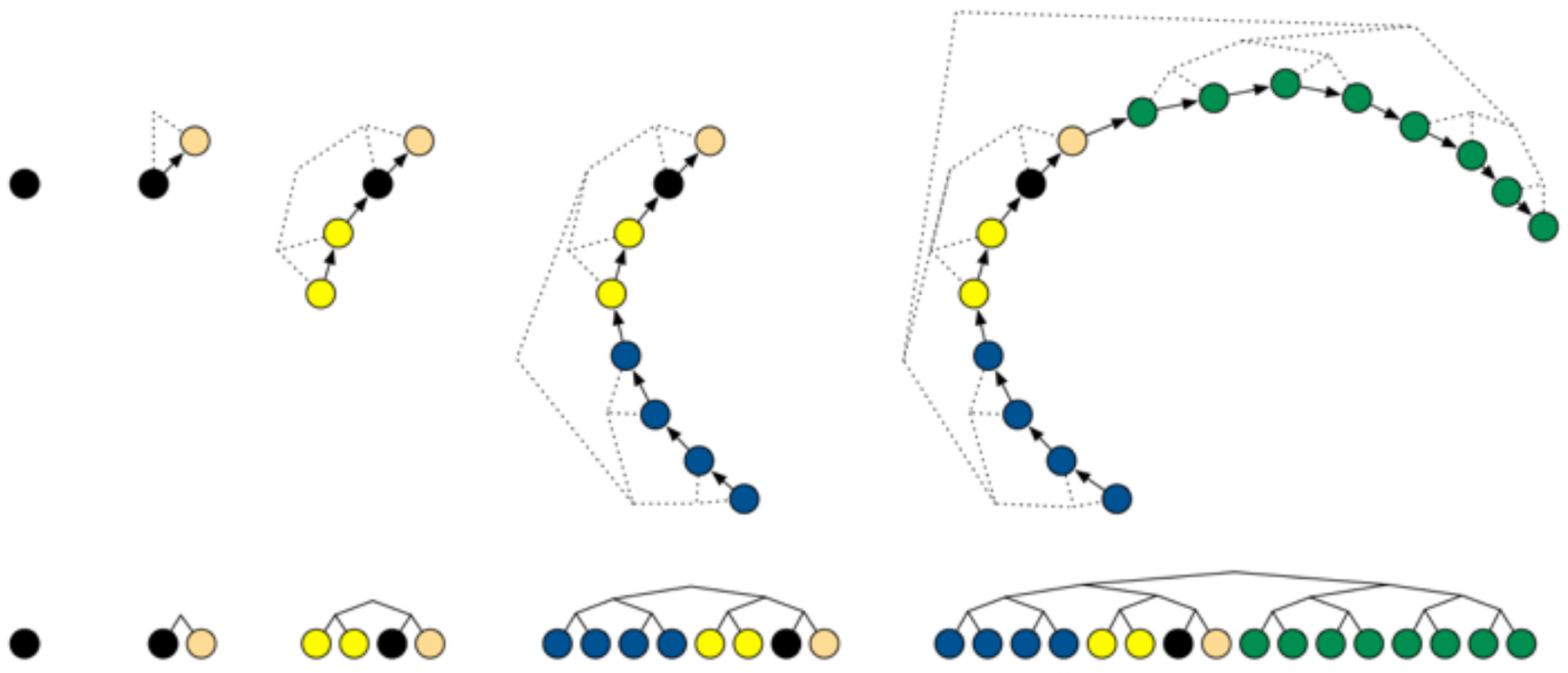
- Use chain rule to break down derivatives until you hit a primitive —> *backpropagation / automatic differentiation*

- 2018: *many* libraries available that implement this (**`autograd`**, **`pytorch`**, **`theano`**, **`tensorflow`**, …)

Autograd demo

Hamiltonian Monte Carlo: setting the number of steps

- Big issue in HMC that trajectory turns back onto itself
- No U-Turn Sampler (Hoffman & Gelman 2011): automatic way to detect whether trajectory is bending back onto itself and stop leapfrog integration



`stan`

- Modeling framework with NUTS HMC sampler at its core
- Specify model in terms of the modeling language, **`stan`** then takes care of everything else
- Supports a *very* large range of possible models, all through the magic of automatic differentiation
- Can significantly speed-up MCMC sampling of many problems, especially ones with many parameters
- C++ library with wrappers in R, Python, cmdline, …
- Similar more Pythonic package: **`pymc3`**

`stan` demo

MCMC overview

- Metropolis-Hastings: simple to implement, need to pick proposal distribution, need to monitor acceptance fraction

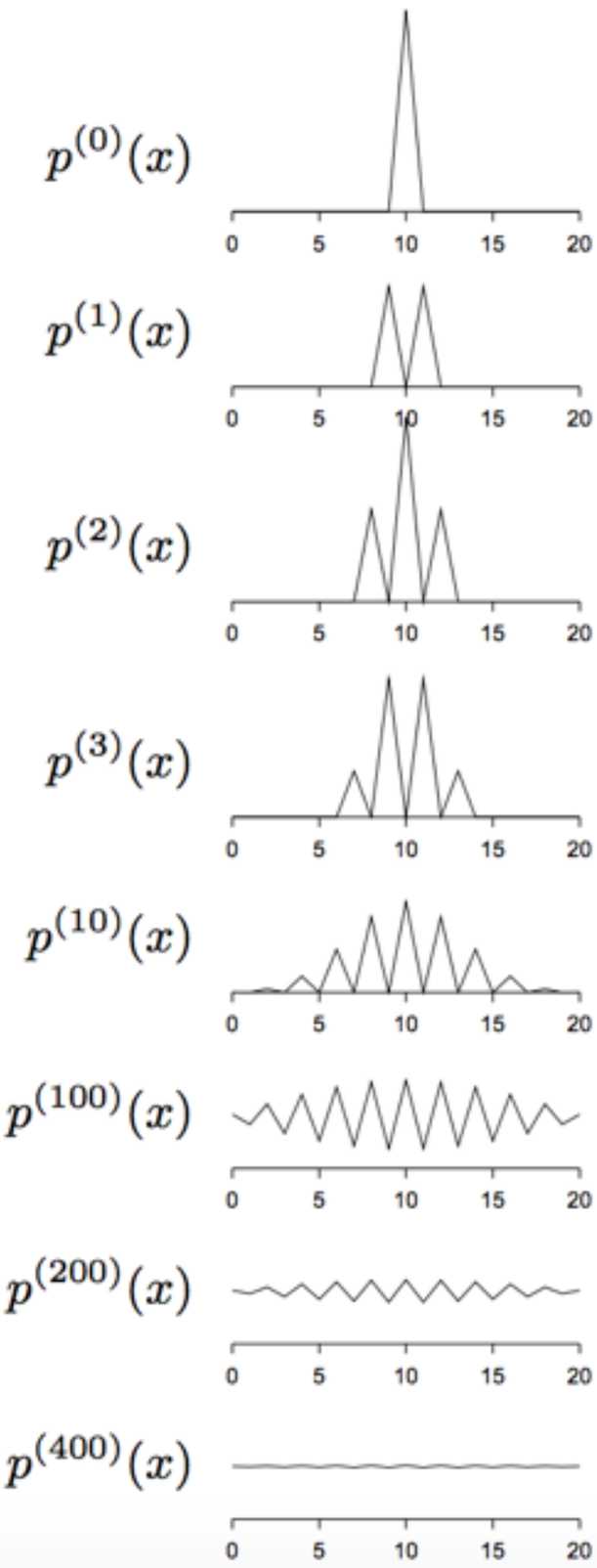
- Gibbs sampling: Great when (some) conditional probabilities are simple

- emcee: Insensitive to step size, so good go-to methods that don't require much supervision; good python implementation of ensemble sampler *emcee* (http://dan.iel.fm/emcee)

- Hamiltonian Monte Carlo: far more efficient exploration of parameter space, viable through multiple software packages today

MCMC: burn-in

- All MCMC algorithms need to ‘burn in’: Takes some number of steps to reach the target distribution p(x)
- Need to monitor convergence to p(x) somehow:
  - Can look at ln[p(x)] and how it evolves over time —> should start randomly oscillating around typical value
  - Can compute desired integrals (e.g., mean) and see when their value stops changing
  - Can run different chains and look at variance between those chains
- Determine when your chain has burned-in, remove everything up to that point; samples are what follows



MCMC: auto-correlation time

- Samples in Markov Chain are correlated, because each value depends on the previous value

- This is okay when computing summaries of the PDF [e.g., $\int d\theta\ p(\theta)\ f(\theta)$] in that this does not introduce *bias*, but it does mean that the uncertainty in the summary does not decrease as $1/\sqrt{N}$

- Can compute the autocorrelation function of your samples: $A(\tau) = \langle x_i\ x_{i+\tau} \rangle$ and determine typical value of τ for autocorrelation to become zero —> *auto-correlation time* τ

- N/τ ~ # of independent samples in your chain

- Can discard non-independent samples; most summaries can be computed using very few independent samples (~12)

Non-parametric ways to estimate uncertainties: Bootstrap and Jackknife

Non-parametric methods

- Bayesian inference requires good knowledge of model, data uncertainties, and everything else involved in going from the model —> data

- Bootstrap and jackknife attempt to quantify uncertainty from the distribution of data itself

- Bootstrap (not the web framework…): data $\{x_i\}$ sampled from some distribution p(x), estimate as

  $p(x) \sim 1/N \times \Sigma_i \delta(x-x_i)$

- Sample new data sets from this estimate of p(x)

Bootstrap

- Suppose you want to know the standard deviation of a set of N data $\{x_i\}$ —> unbiased estimator

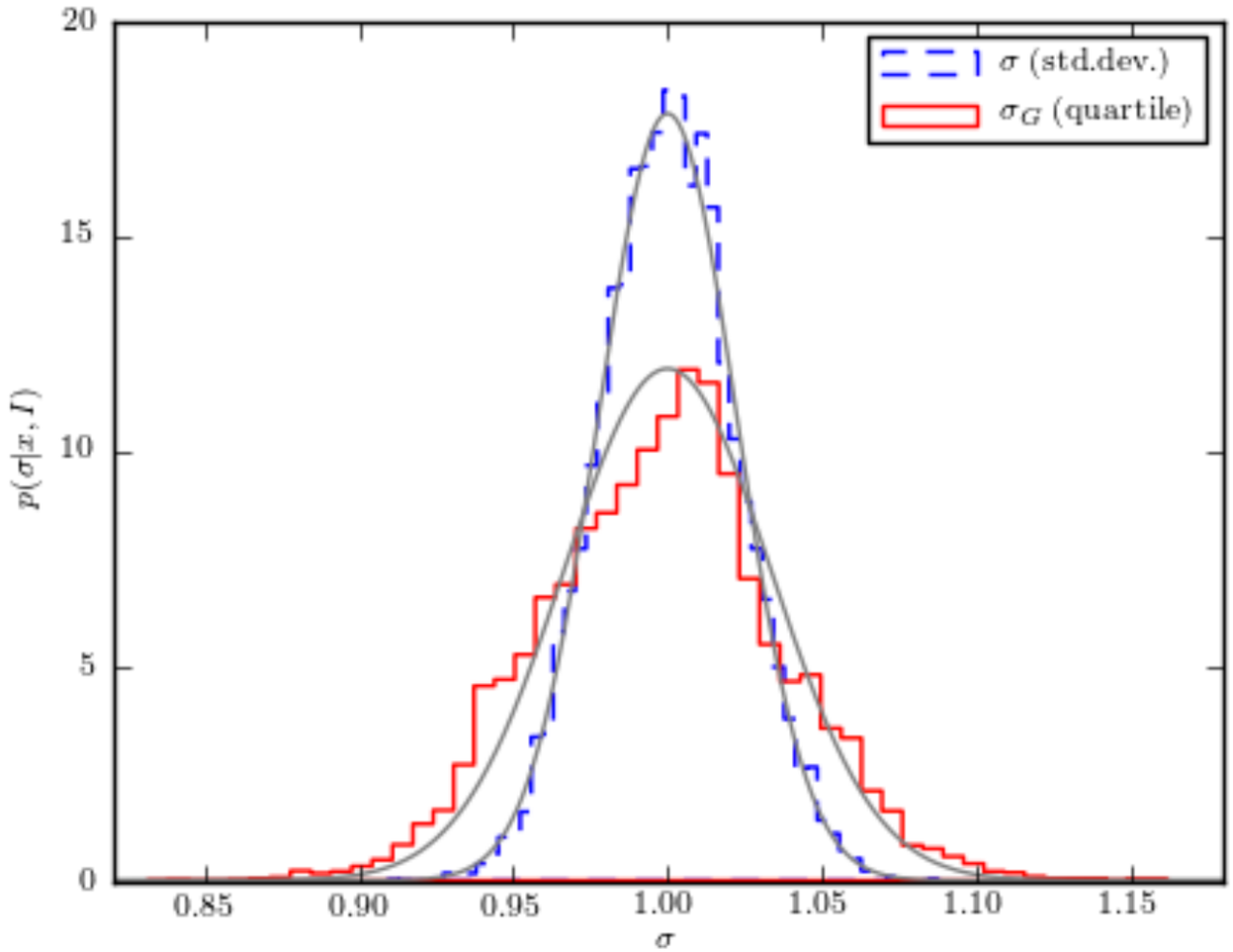
  $\sigma^2 = 1/[N-1]\Sigma_i [x_i-\langle x_i\rangle]^2$

  What is its uncertainty?

- Bootstrap: sample new data points from $p(x) \sim 1/N \times \Sigma_i \delta(x-x_i)$ —> sample N 'new' data points from the original set with replacement (i.e., can sample the same one twice)

- Compute σ^2 for each resampling —> distribution of these σ^2 is the uncertainty distribution

Bootstrap



Ivezic et al. (2014)

Jackknife

- Rather than sampling with replacement, make N new data sets by leaving out 1 data point at a time

- So $\{x_1,x_2,x_3,\ldots\}$, $\{x_0,x_2,x_3,\ldots\}$, $\{x_0,x_1,x_3,\ldots\}$, …

- Compute estimator θ for each subsample, θ_{-i}

- Uncertainty in estimator:

$$\sigma^2 = \frac{N-1}{N}\sum_i(\theta_{-i} - \theta_{\rm all})^2$$

where

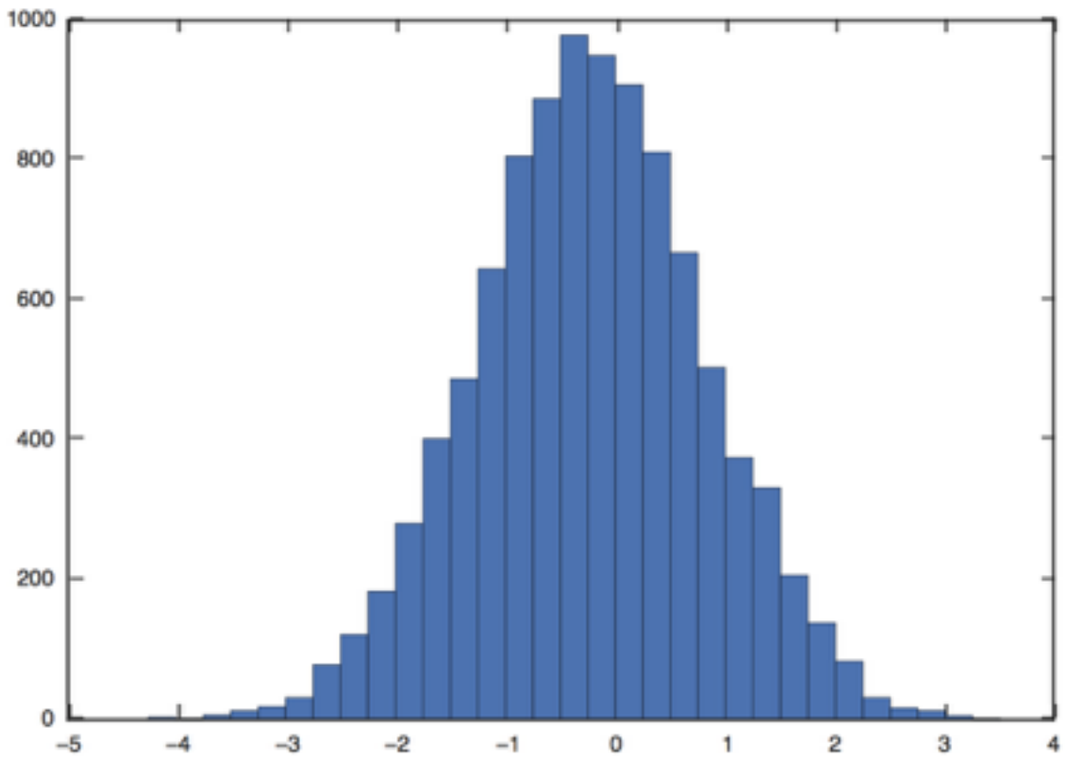
$$\theta_{\rm all} = \frac{1}{N}\sum_i \theta_{-i}$$

Robust against underestimating one's errors and correlated errors

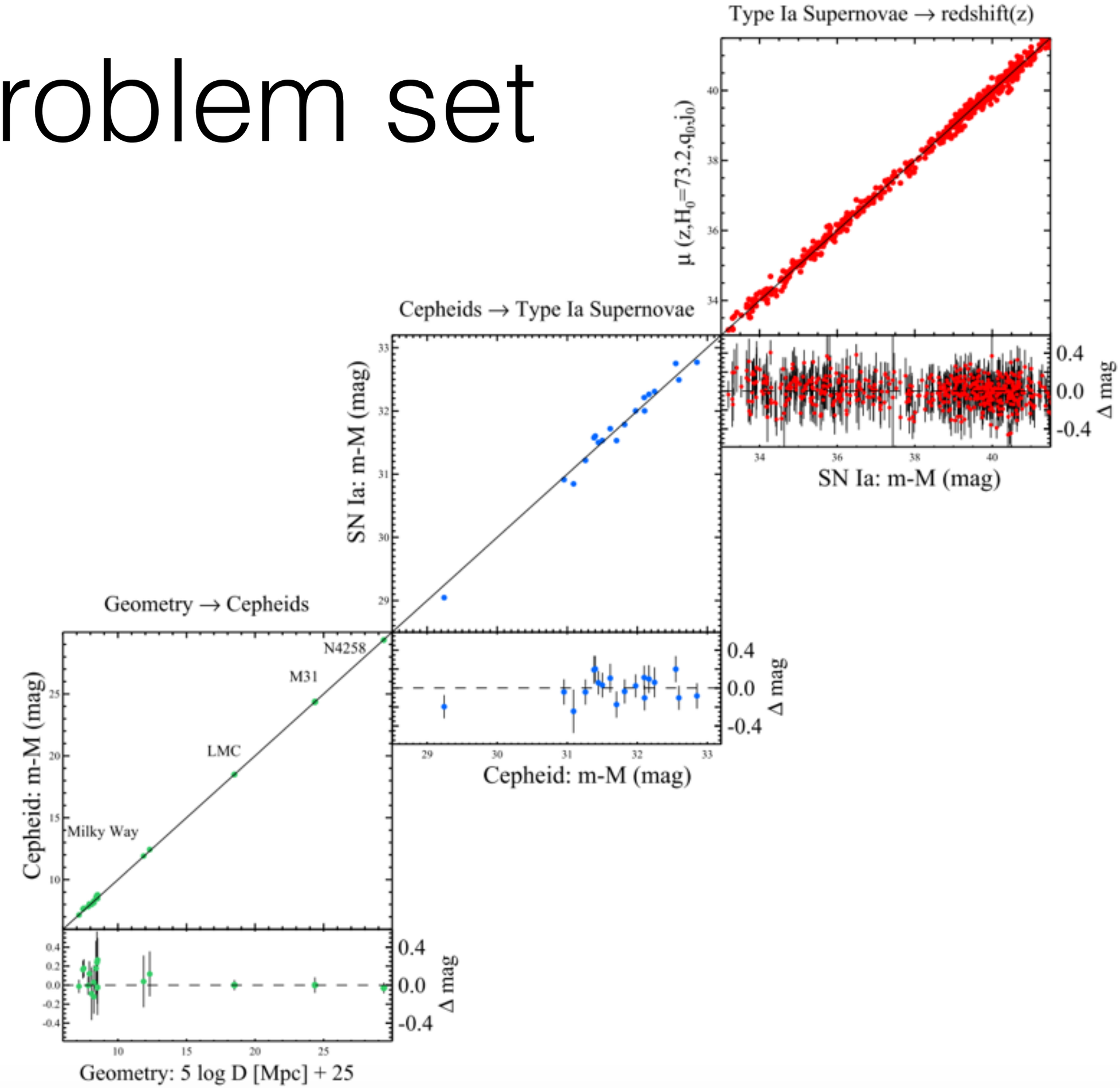
- Suppose you want to know the mean of a set of data that you think have errors of 2, but really have errors of 10

- 100 data points: Would assign mean error $2/\sqrt{100} = 0.2$; but real error is $10/\sqrt{100} = 1$

- Bootstrap: error=1



Problem set


Type Ia Supernovae → redshift(z)
μ (z,H0=73.2,q0,j0)
Cepheids → Type Ia Supernovae
SN Ia: m-M (mag)
Δ mag
Geometry → Cepheids
N4258
M31
LMC
Milky Way
Cepheid: m-M (mag)
Geometry: 5 log D [Mpc] + 25