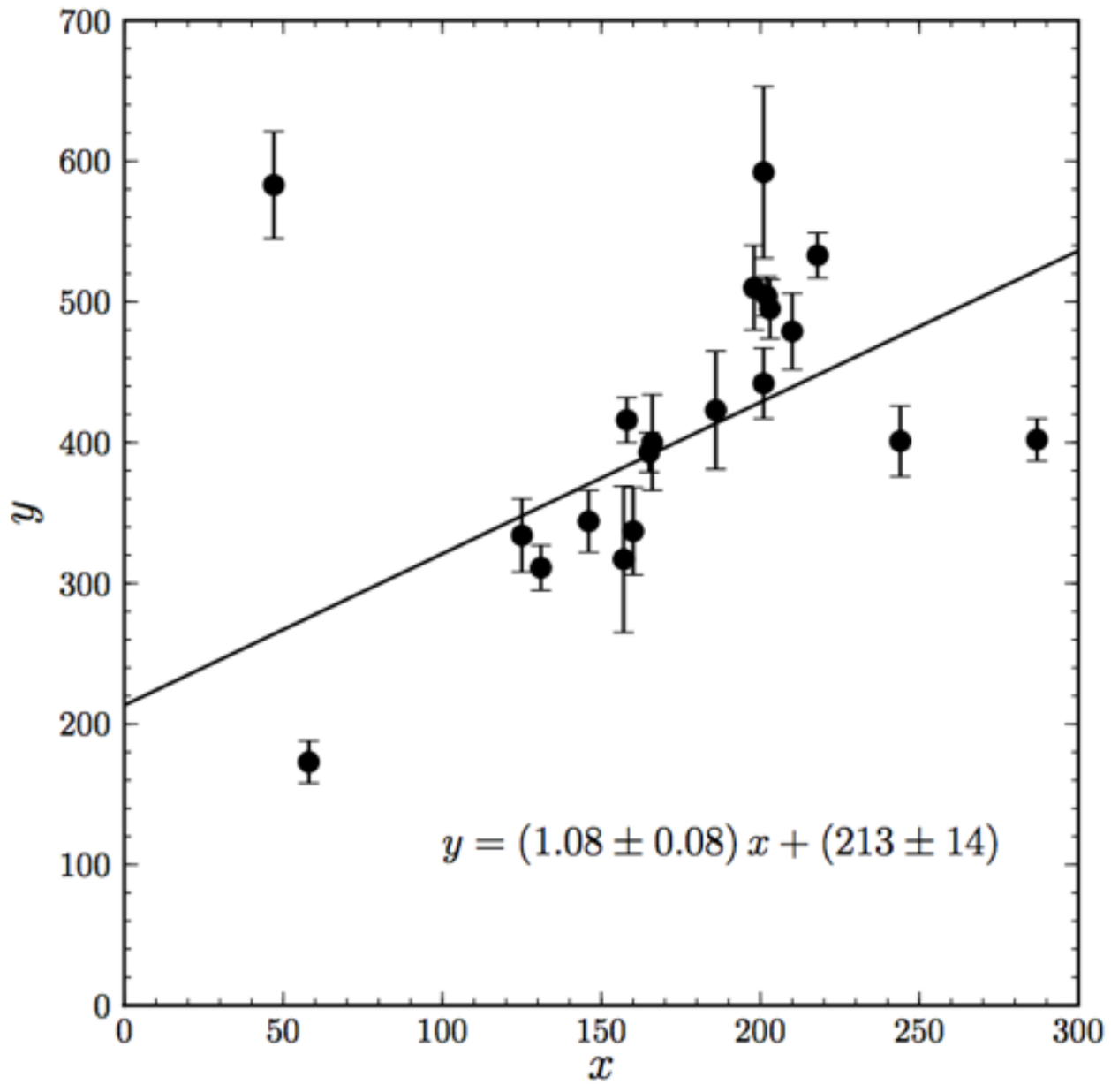
Statistics and Inference in Astrophysics

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*

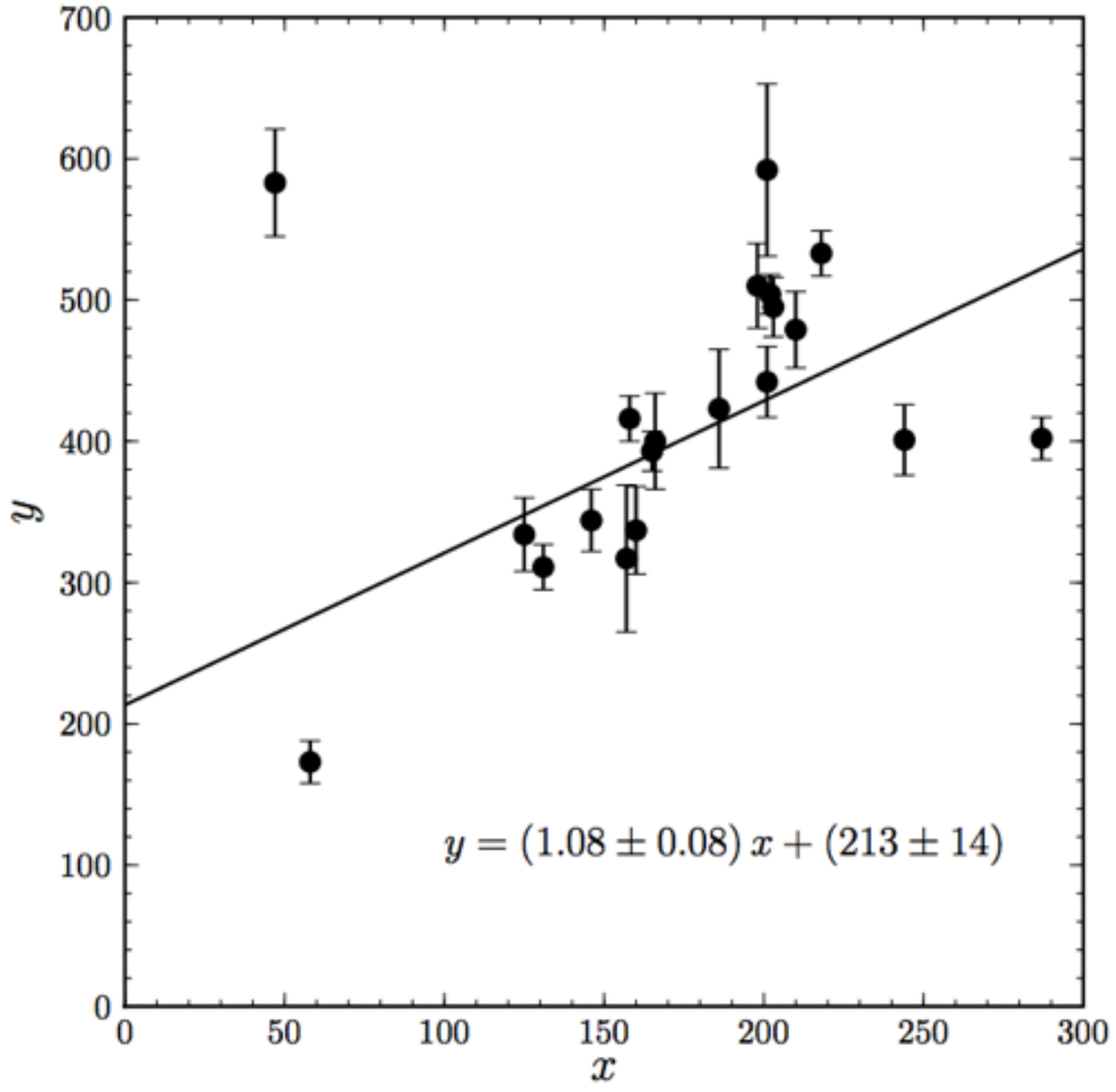
- Likelihood, single point: $p(y_i|m,b,x_i,\sigma_{y,i}) = N(y_i|mx_i+b,\sigma^2_{y,i})$

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

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

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

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!

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

  $\int d\theta\, f(\theta) \sim V \times 1/N \times \Sigma_i f(\theta_i)$

  where

  $V = [\int d\theta]$,

  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

$$\int d\theta\, f(\theta) = \int d\theta\, q(\theta)\, [f(\theta)/q(\theta)]$$
$$\sim V_q \times 1/N \times \Sigma_i\, f(\theta_i)/q(\theta_i)$$

where

$V_q = [\int d\theta\, q(\theta)]$,

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

  $\int d\theta\, p(\theta)\, f(\theta) = 1/N \times \Sigma_i\, f(\theta_i)$

  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 $p(\theta)$!

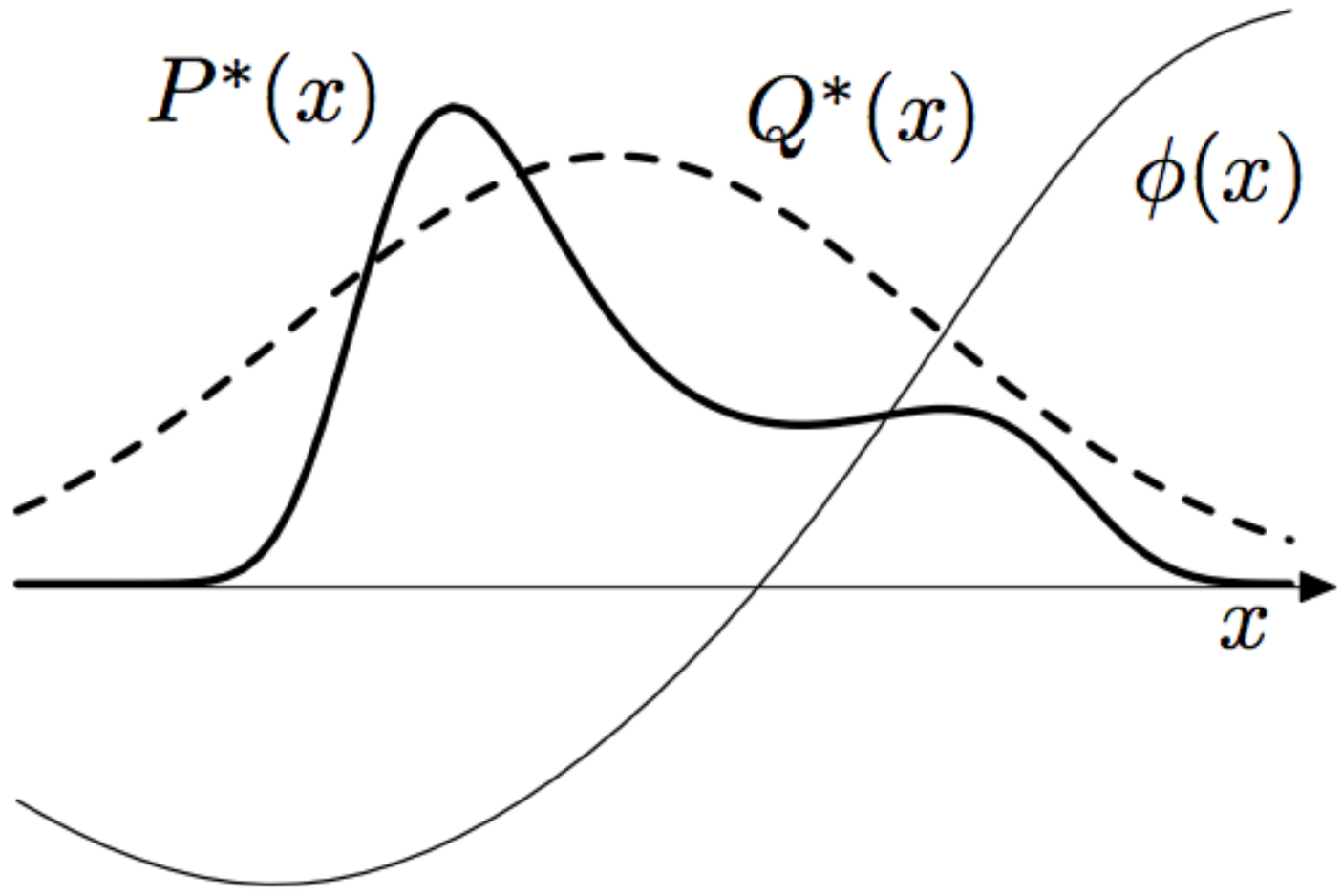
Importance sampling

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

$$\int d\theta\, p(\theta) f(\theta) = \int d\theta\, q(\theta) p(\theta)/q(\theta) f(\theta)$$
$$\sim 1/N \times \Sigma_i f(\theta_i) \times p(\theta_i)/q(\theta_i)$$

- The $p(\theta_i)/q(\theta_i)$ are known as the *importance* weights
- They re-weight the importance of each sample
- Works well if $q(\theta)$ is close to $p(\theta)$, otherwise introduces large variance: think about what happens when $q(\theta)$ is small when $p(\theta)$ 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 $\int d\theta\ p_{you}(\theta|data)\ f(\theta) = \int d\theta\ p(data|\theta)\ p_{you}(\theta)\ f(\theta)$

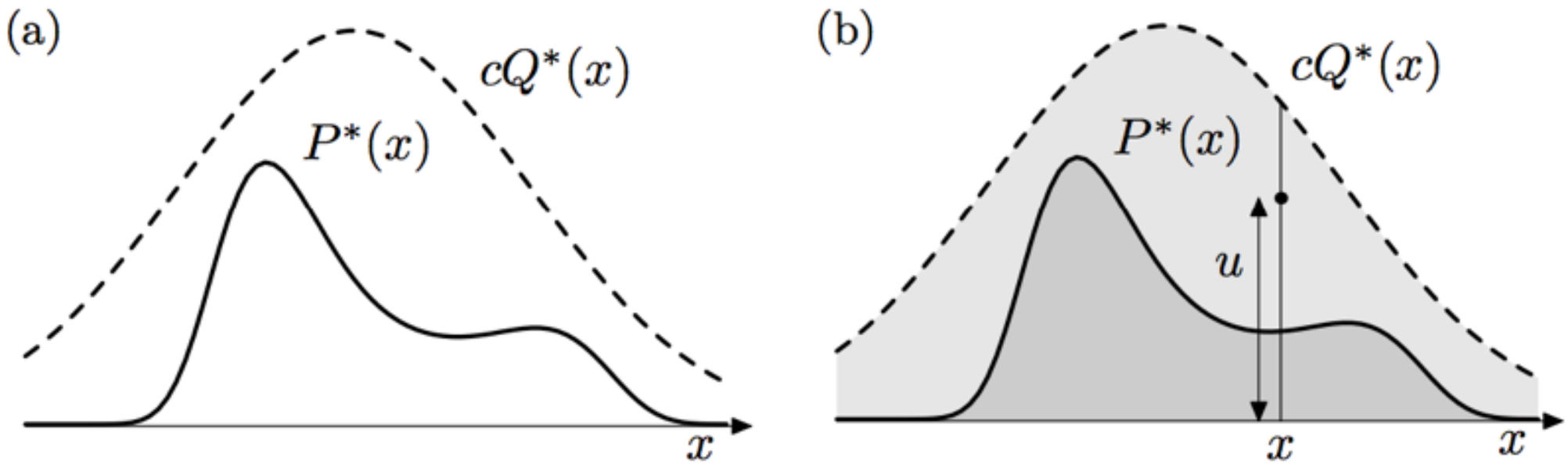
  But you have samples θ_i from
  $p_{not\ you}(\theta|data) = p(data|\theta)\ p_{not\ you}(\theta)$

- Can do

  $\int d\theta\ p_{you}(\theta|data)\ f(\theta) = 1/N \times \Sigma_i\ f(\theta_i) \times\ p_{you}(\theta_i)/p_{not\ you}(\theta_i)$

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

Rejection sampling

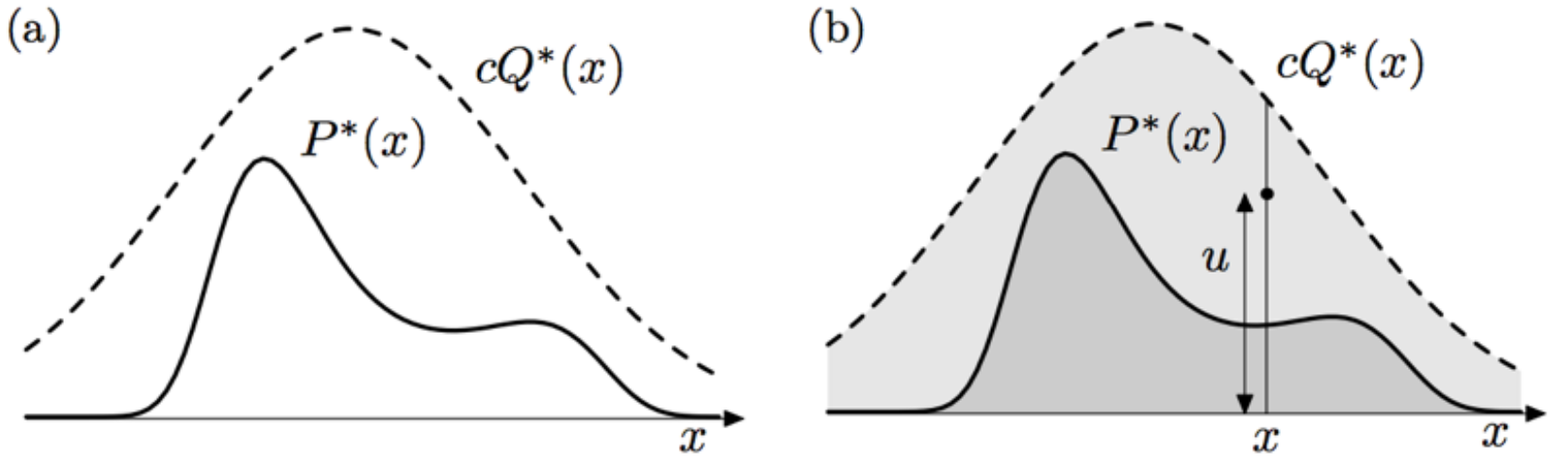


Mackay (2003)

- Imagine binning p(x), then have to sample P(x) = p(x) dx —> can sample p(x) by uniformly sampling the area under the p(x) curve

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)

- Techniques like adaptive-rejection sampling for log-concave distributions iteratively build up tight hull around p(x) that allows efficient sampling (implemented in *galpy*!)

- 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)

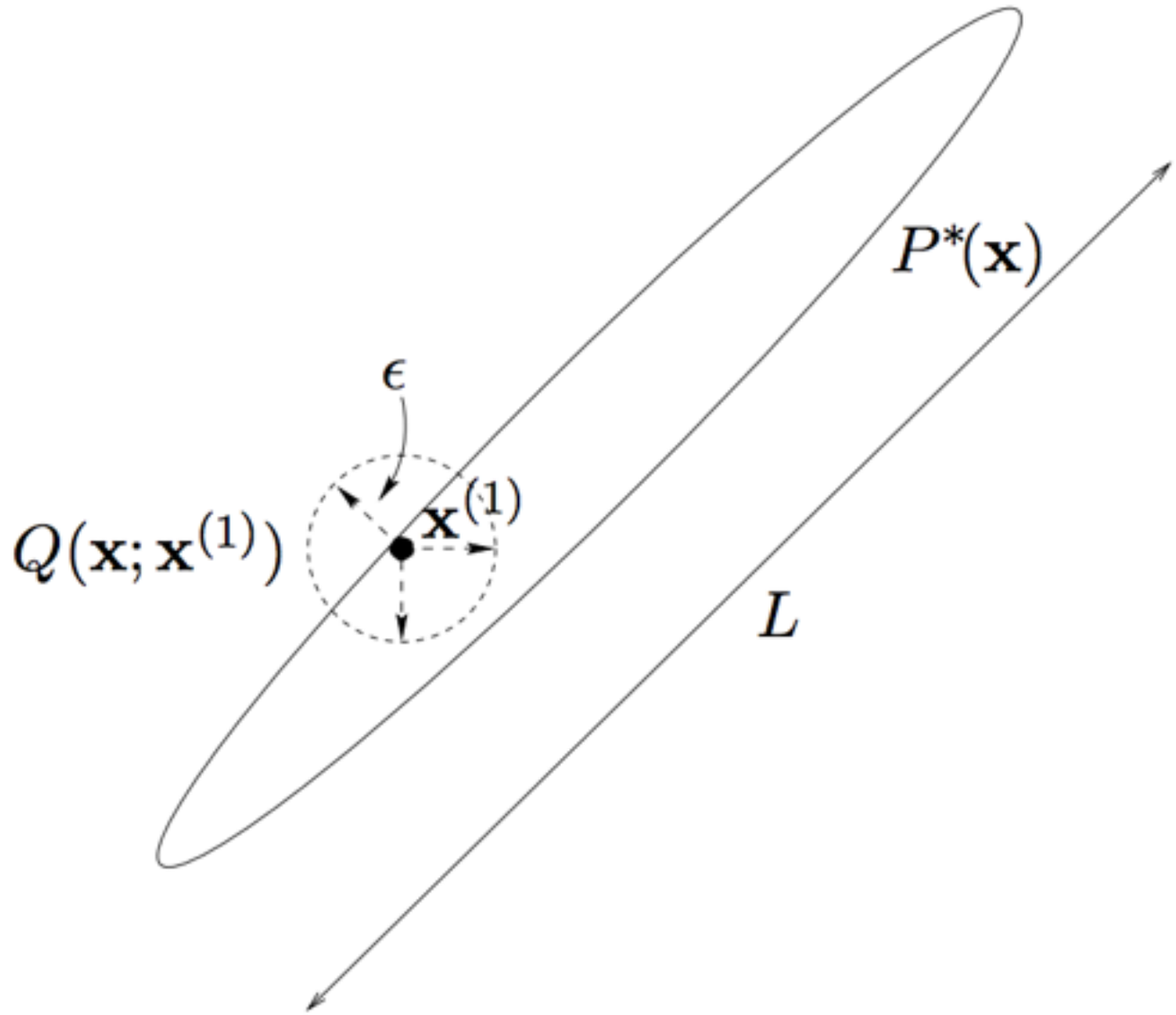
Metropolis-Hastings

- 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: $x_{i+1} = x_i$

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])

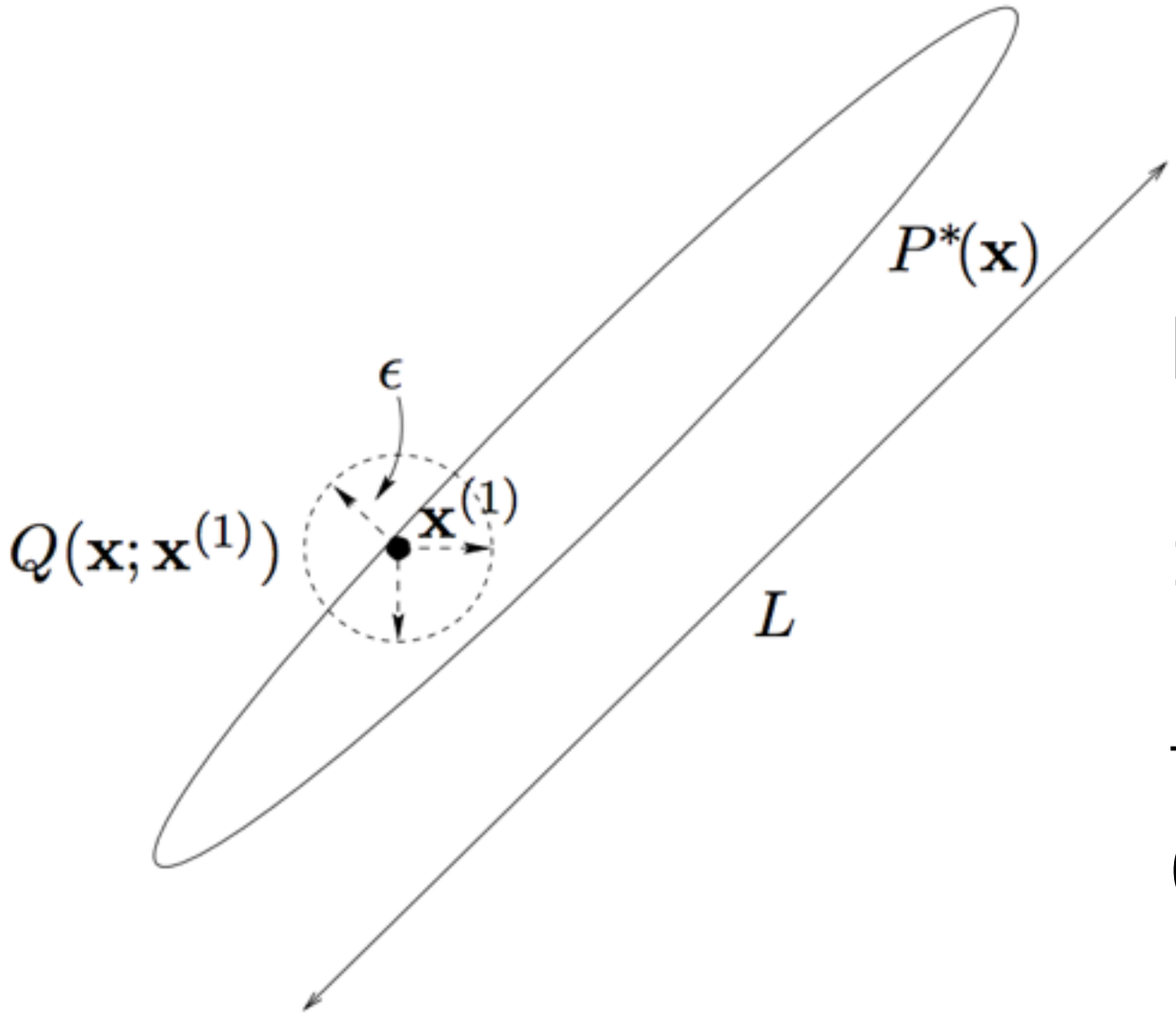
  Set it too large: jump to regions with low p(x) —> reject

  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:

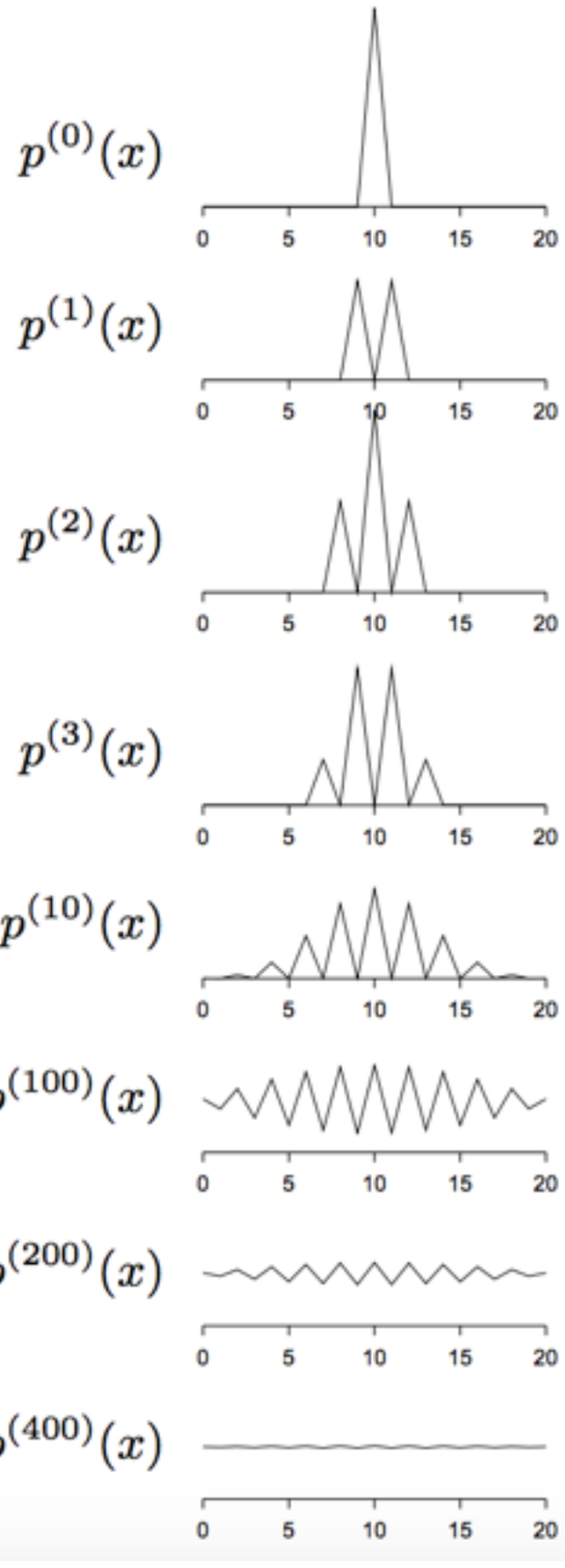
- p(x) should be an invariant distribution of the Markov Chain:

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

- 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

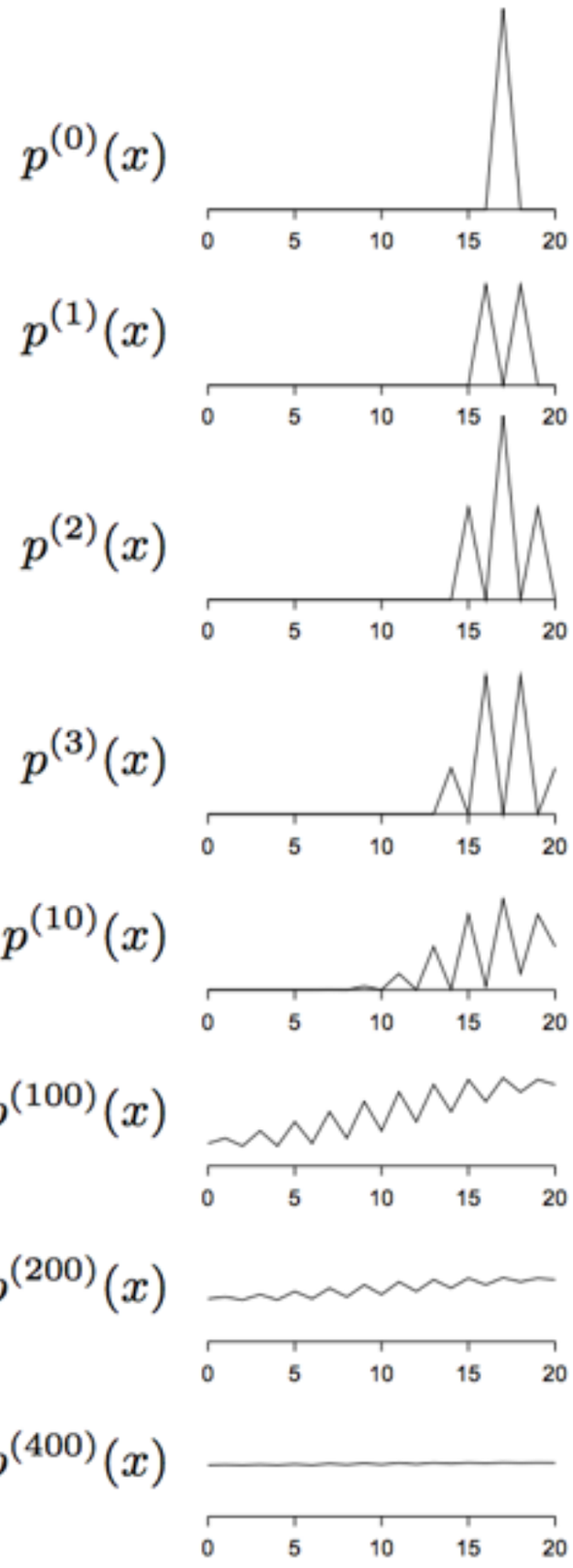
$$\mathbf{T} = \begin{bmatrix} 1/2 & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & 1/2 & . & 1/2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



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)

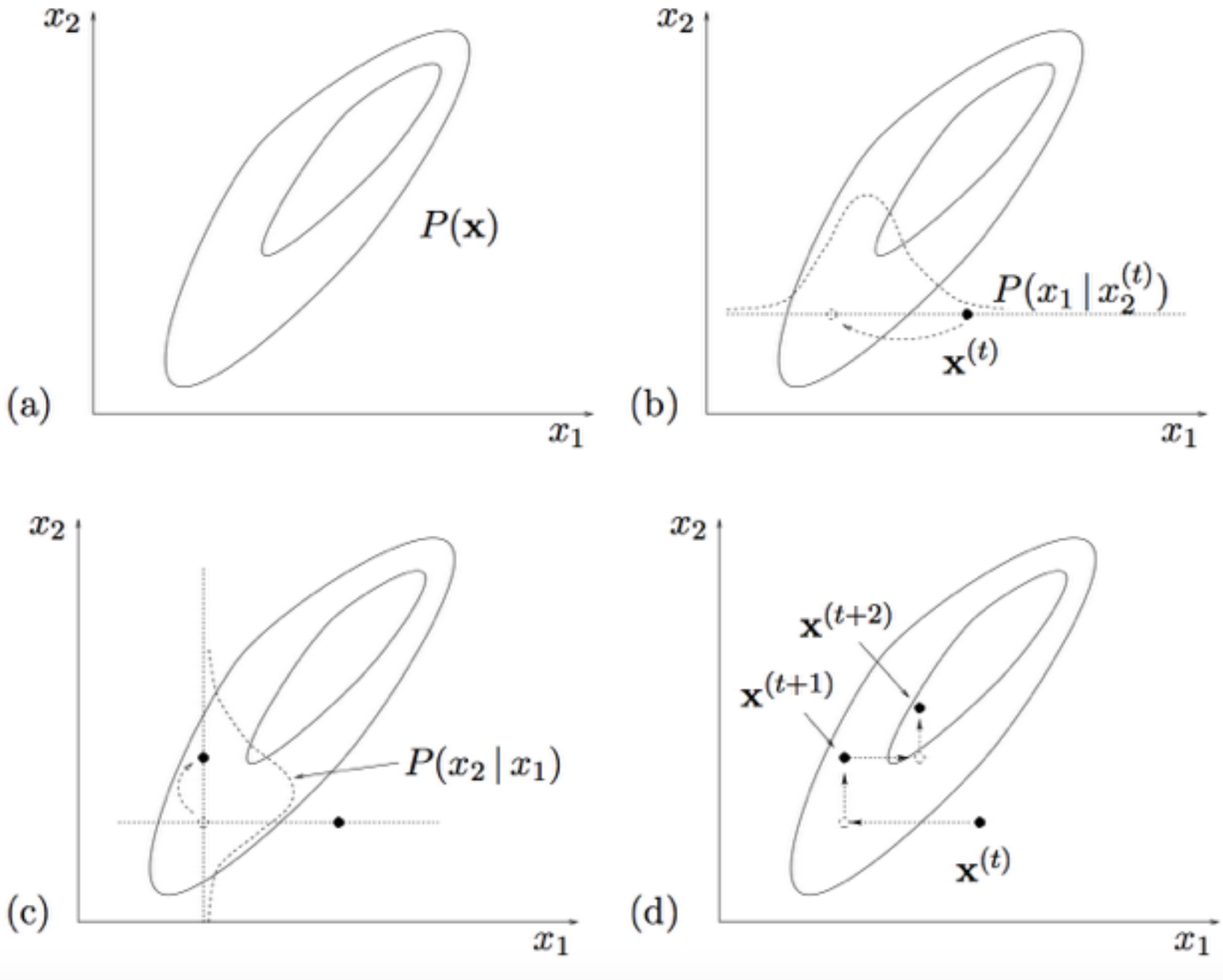
- Sample: Starting at (x_i,y_i)
    1. x_{i+1} from $p(x|y_i)$
    2. y_{i+1} from $p(y|x_{i+1})$
    3. New (x_{i+1},y_{i+1})

- 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

Samplers with less dependence on the step size: Slice sampling

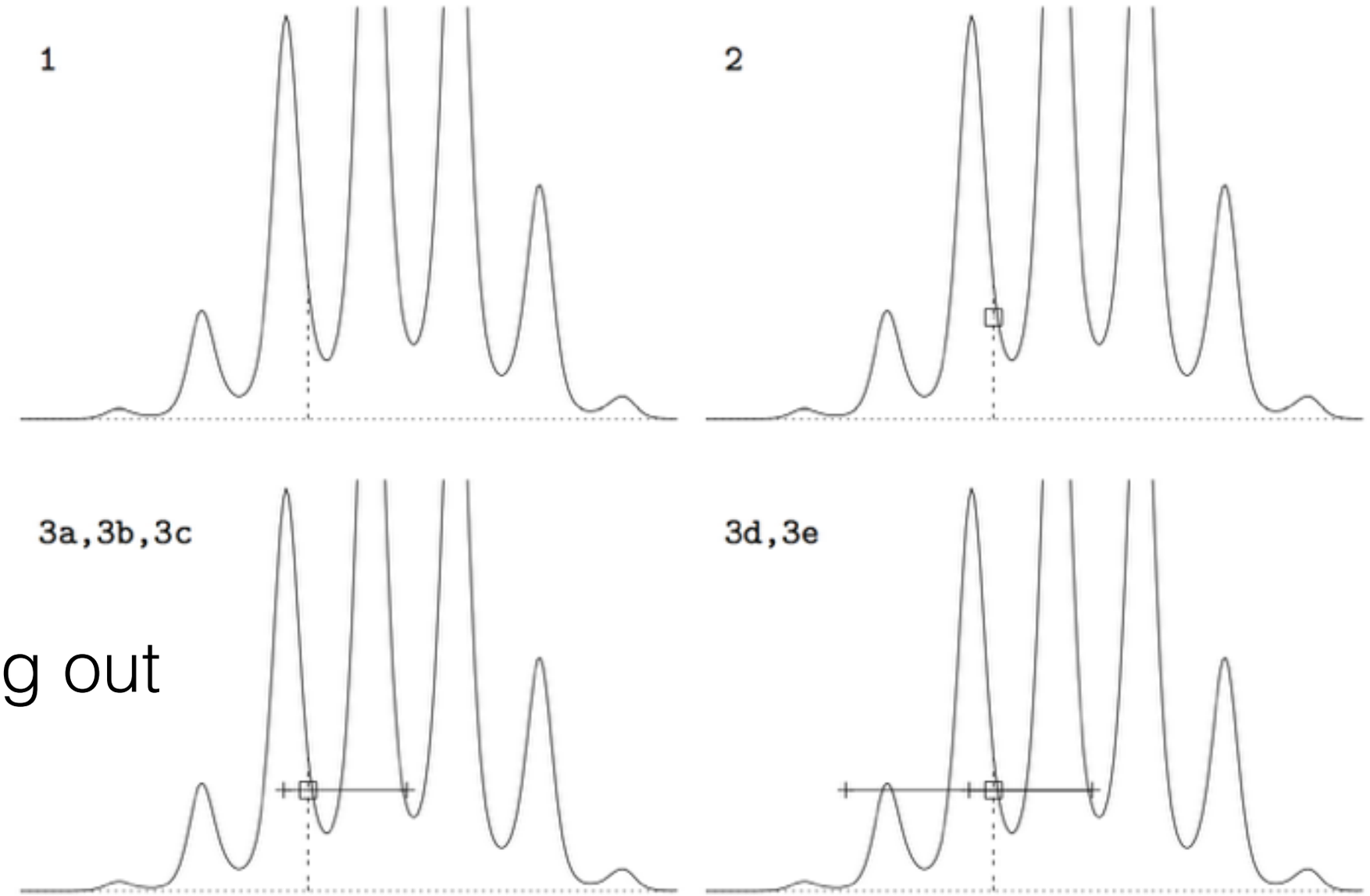
- Similar to rejection sampling and Metropolis-Hastings

- Like rejection sampling, samples the area below p(x) uniformly, but without bounding function

- Algorithm: starting from x_i:
    1. evaluate $p(x_i)$
    2. Draw uniform *u* from [0, $p(x_i)$]
    3. Create interval (x_{low},x_{high}) that encloses x_i (*stepping out*)
    4. Loop through:
        * Draw x' uniformly from [x_{low},x_{high}]
        * if p(x') > *u*: break and accept x'
        * else: modify (x_{low},x_{high}) (*shrinking*)

- Interval creation and modification by stepping out and shrinking

Slice sampling

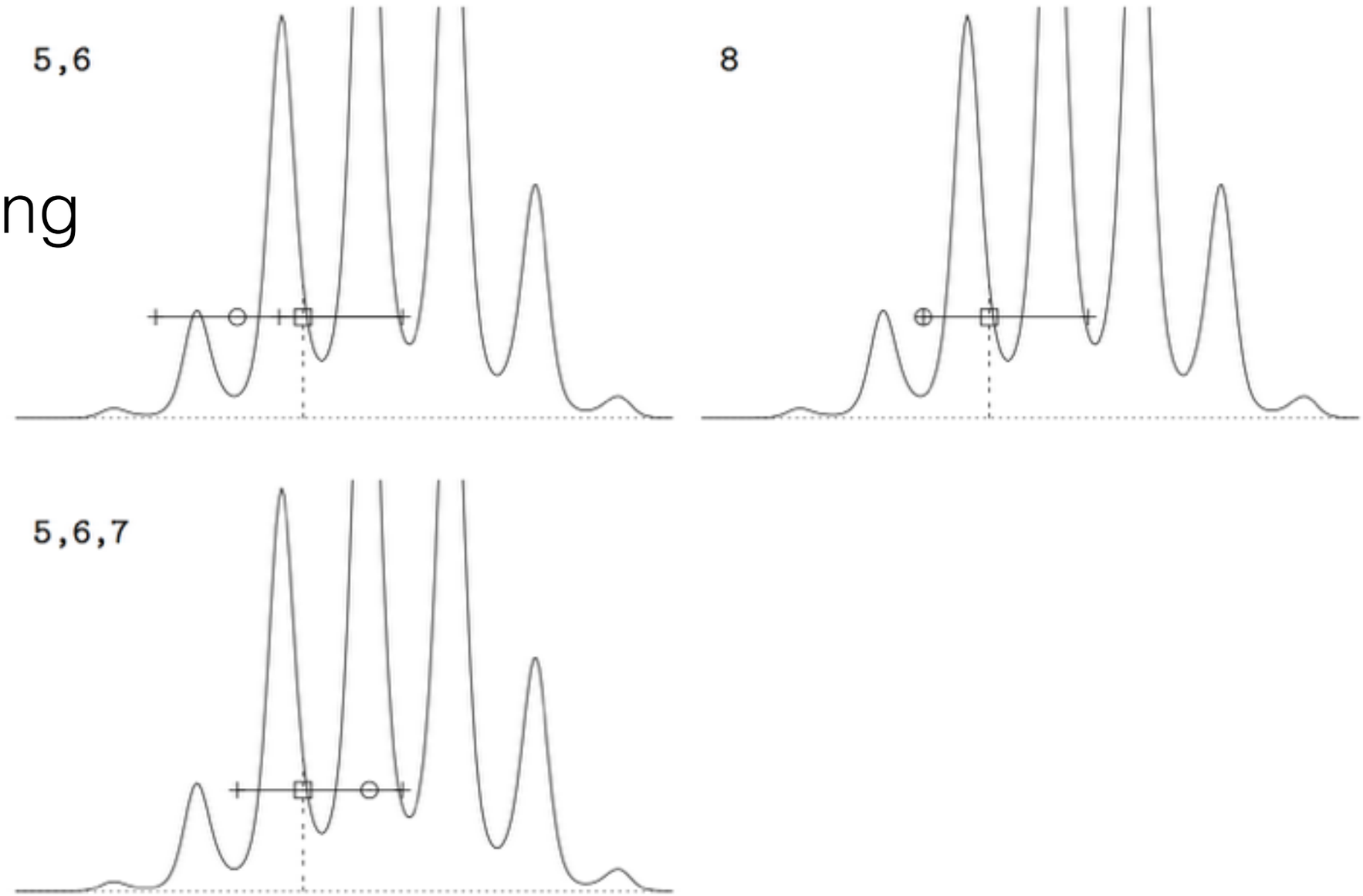


Stepping out

Mackay (2003)

Slice sampling

Shrinking



Mackay (2003)

Slice sampling: advantages

- Requires step size to set the stepping-out step, but algorithm automatically adjust for bad choices (L ~ width of PDF):

  Step too small: algorithm will take many steps to step-out, but eventually end up O(L) away

  Step too large: algorithm steps out quickly, but needs to shrink in many steps

- Every sample accepted! No need to check acceptance fraction (but want to monitor # of stepping-out and # shrinking iterations)

- Used to be my go-to method, still useful for some problems (e.g., Gaussian processes)

Ensemble samplers

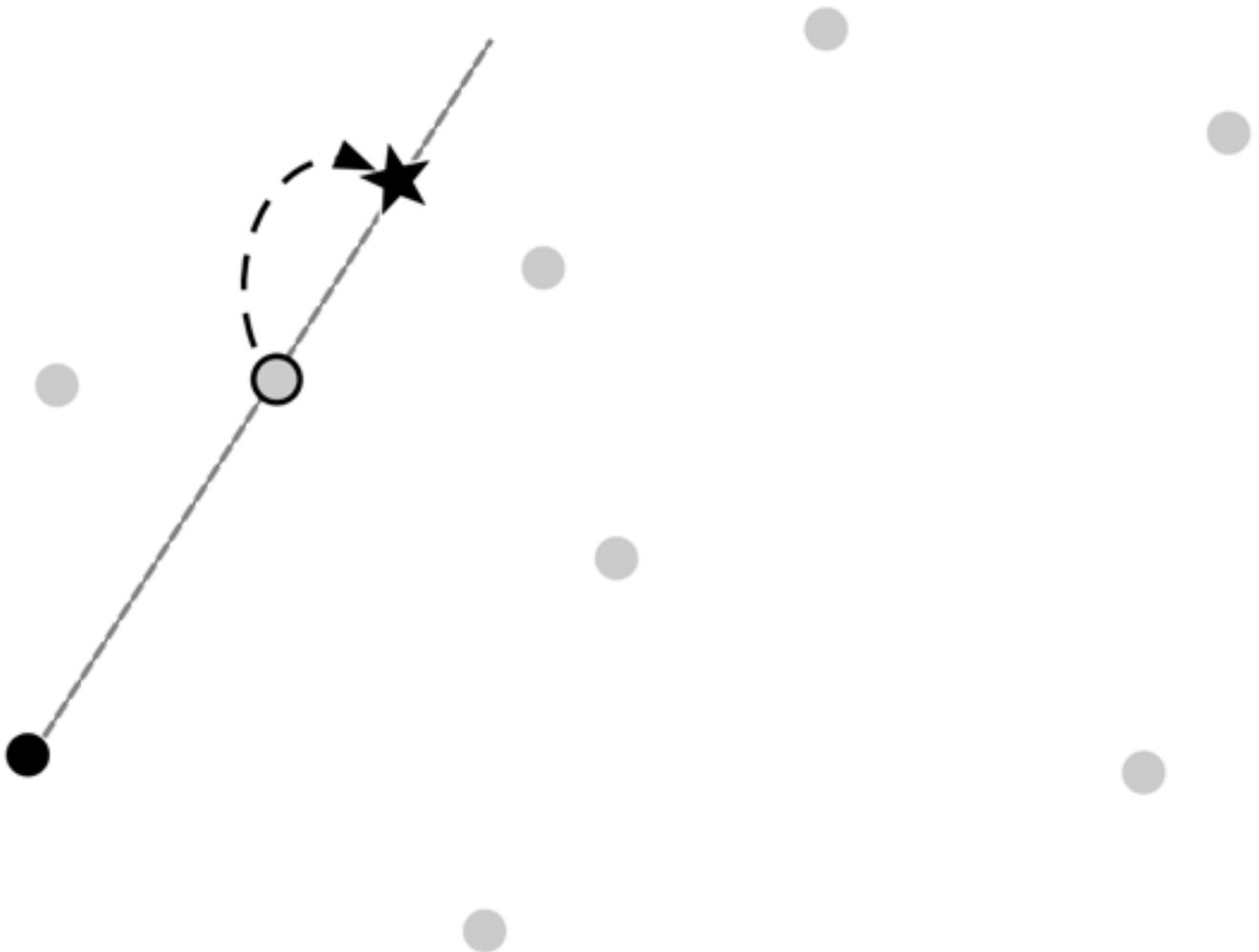
- So far have considered single sample x_i that gets updated

- 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*)

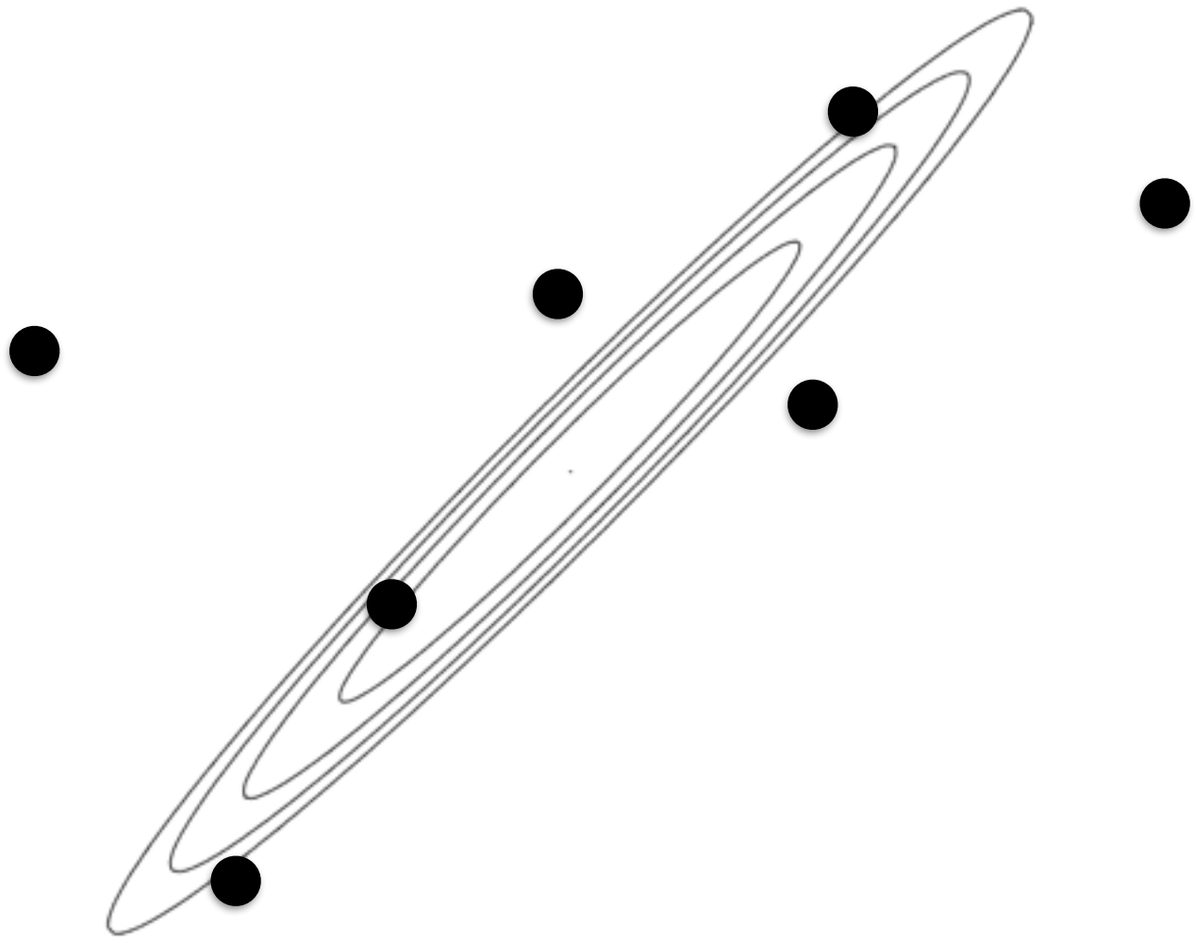
- Variations have different points in the ensemble at different temperatures, …

Affine-invariant sampler (*emcee*)



Goodman & Weare (2009)

Affine-invariant sampler (*emcee*)



Goodman & Weare (2009)

Affine-invariant sampler (*emcee*)

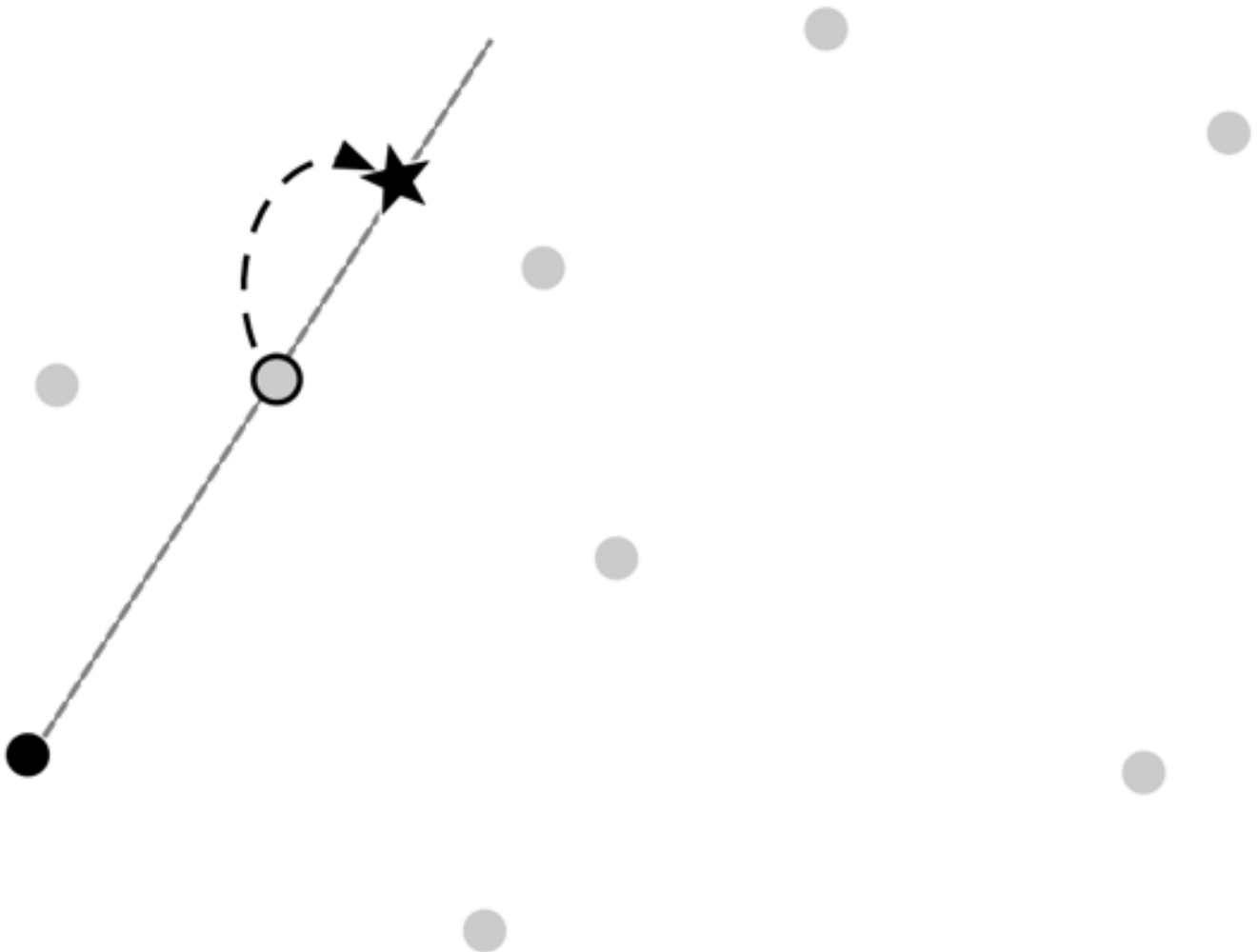
- Each x in $\{x\}_i$ is called a *walker*

- 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

MCMC overview

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

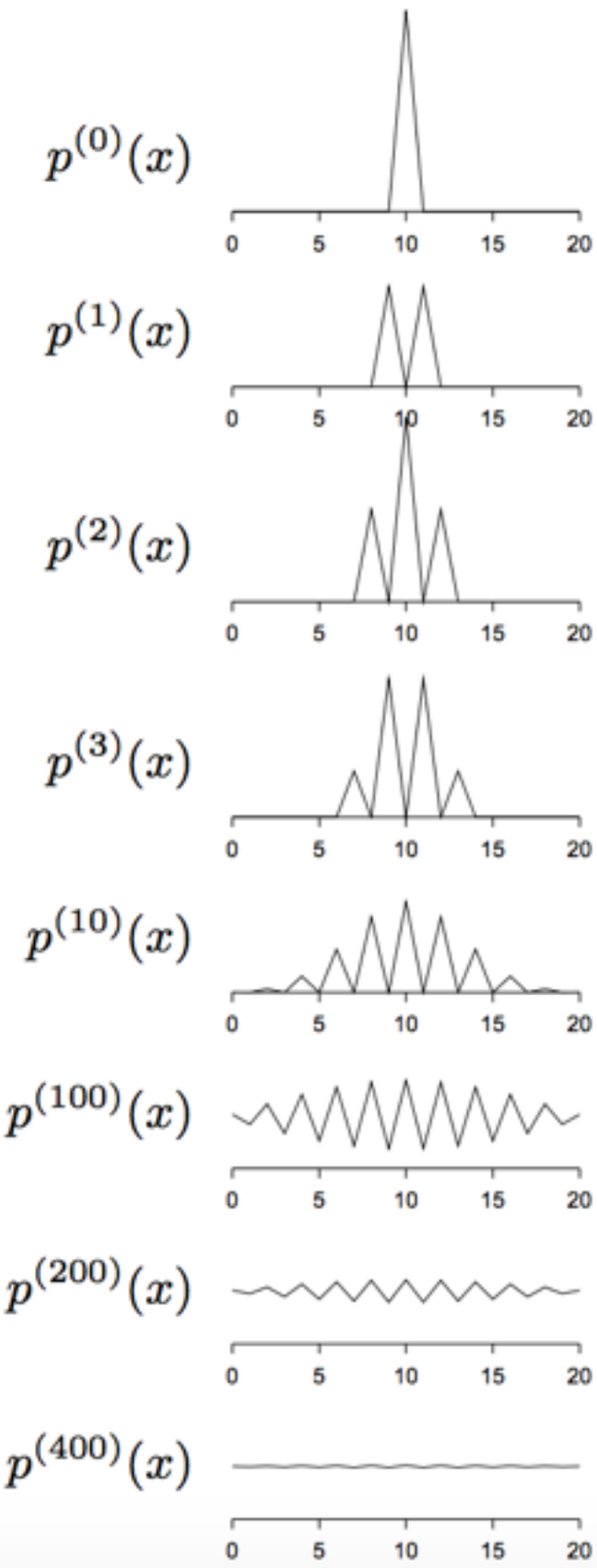
- Gibbs sampling: Great when (some) conditional probabilities are simple

- Slice sampling, 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)

- Not talked about: Nested sampling, Hamiltonian Monte Carlo (uses derivatives of PDF), more complicated ensemble samplers

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

  $p(x) \sim 1/N_{data} \times \Sigma_i \delta(x-x_i)$

- 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

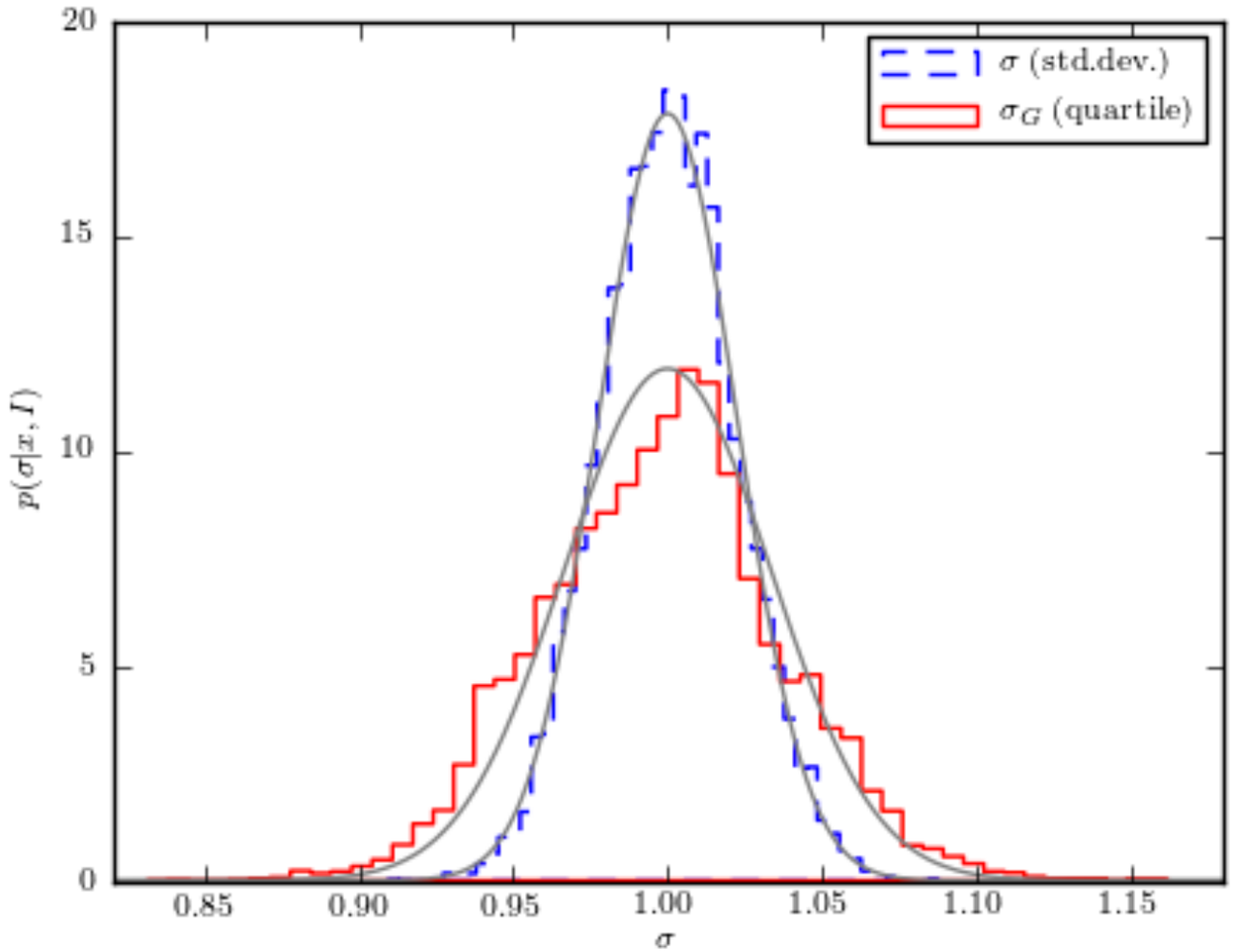
  $\sigma^2 = 1/[N-1]\Sigma_i [x_i-\langle x_i\rangle]^2$

  What is its uncertainty?

- Bootstrap: sample new data points from p(x) ~ 1/Ndata x $\Sigma_i \delta(x-x_i)$ —> 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:

  $\sigma^2 = [N-1]/N \, \Sigma_i \, (\theta_{-i}-\theta_{all})^2$

  where

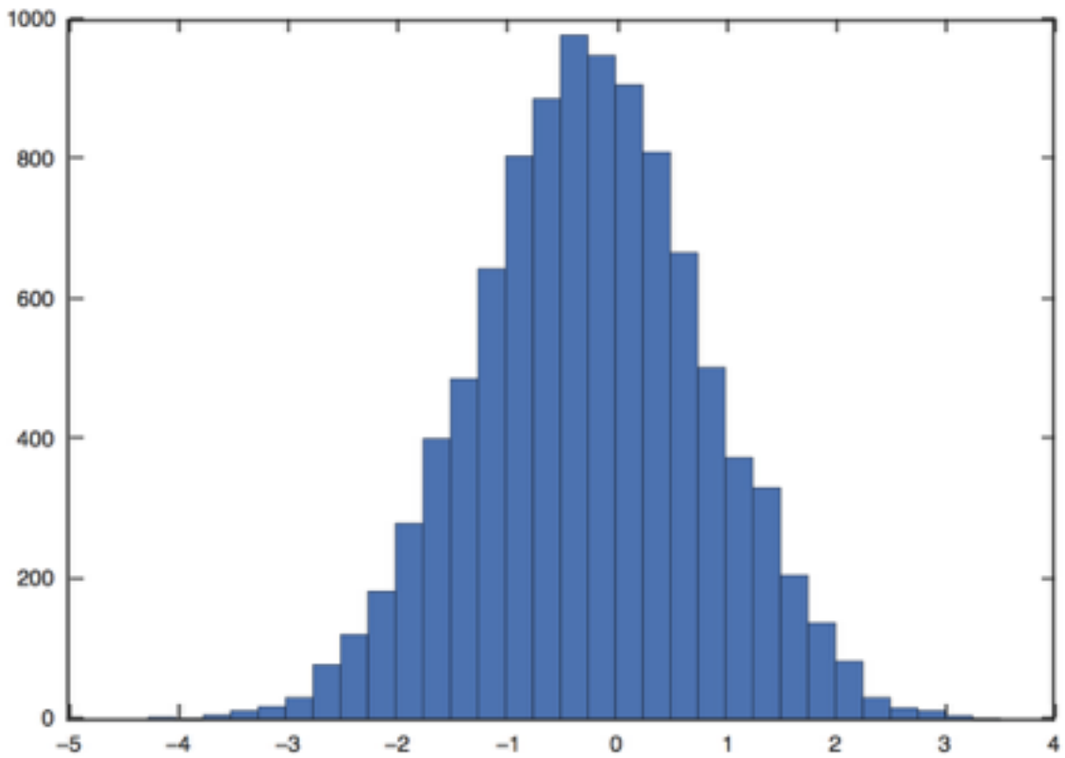
  $\theta_{all} = 1/N \, \Sigma_i \, \theta_{-i}$

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 1

- Four exercises of inference and MCMC
- Due Mar. 31
- Please let me know whether you are taking the course for credit when you submit

Problem set 1

Statistics Mini-course Problem Set 1

Due on Thu. Mar 31

We will do some of the exercises in Hogg, Bovy, & Lang (HBL; 2010) (`1008.4686`), with some slight variations. You should solve these exercises on a computer and the best way to hand in the problem set is as an `ipython notebook`. Rather than sending me the notebook, you can upload it to `GitHub`, which will automatically render the notebook. Rather than starting a repository for a single notebook, you can upload your notebook as a `gist`, which are version-controlled snippets of code.

If you want to upload your notebook as a gist from the command-line, you can use the package at this http URL and use it as follows. Log into your `GitHub` account:

```
gist --login
```

and then upload your notebook `statminicourse_2016_PS1_YOURNAME.ipynb` as

```
gist statminicourse_2016_PS1_YOURNAME.ipynb
```

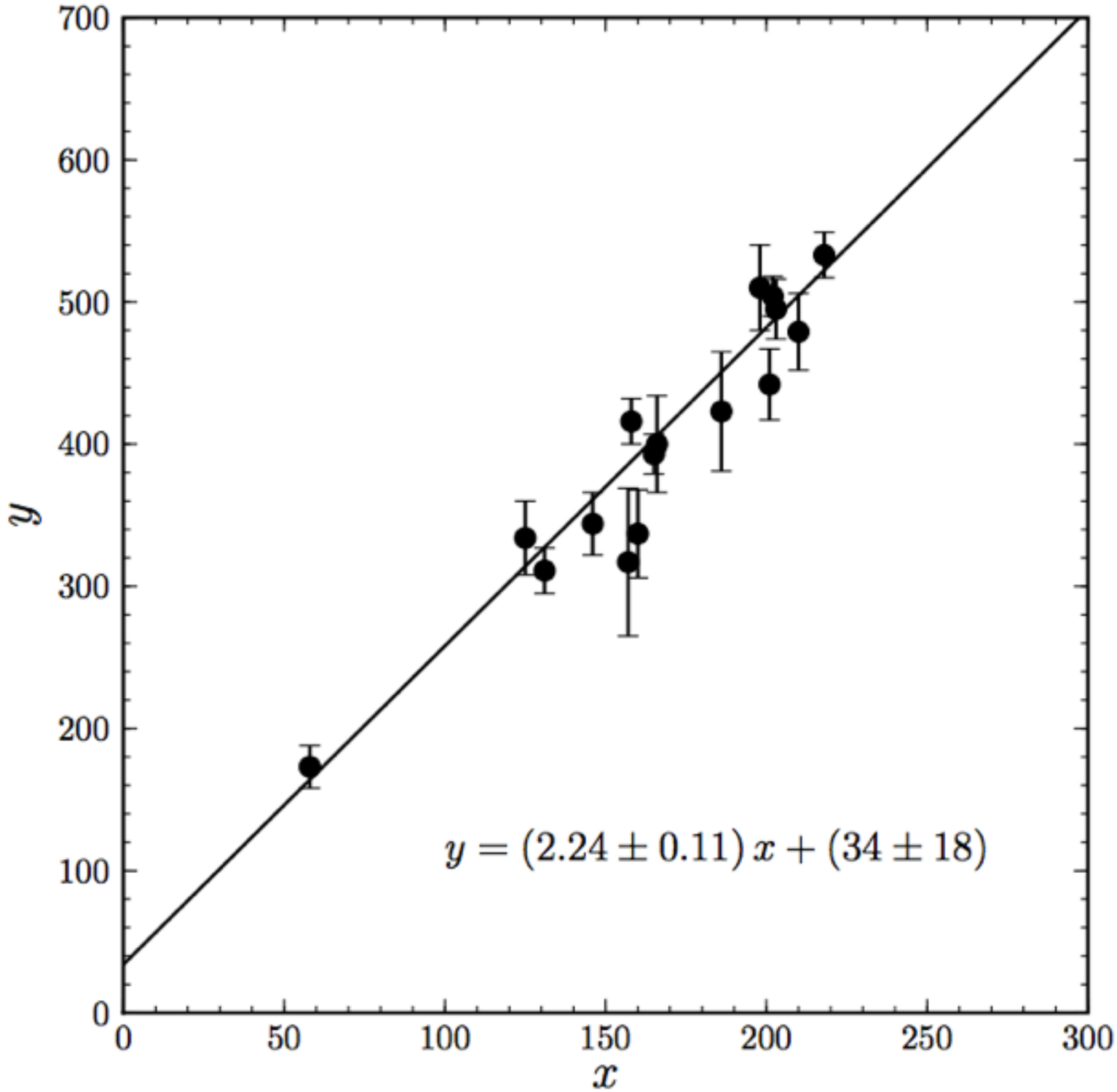
If you want to make further changes, you can clone your gist in a separate directory and use it as you would any other git repository.

Problem 1: Do exercise 1 in BHL.

Problem 2: Do exercise 1, but assuming that the errors σ_y of neighboring data points in x are correlated with a correlation coefficient of $\rho = 0.5$. E.g., data points 15 and 16 have y measurements whose uncertainty is described by a covariance matrix $\begin{pmatrix} \sigma_{y,15}^2 & 0.5\,\sigma_{y,15}\,\sigma_{y,16} \\ 0.5\,\sigma_{y,15}\,\sigma_{y,16} & \sigma_{y,16}^2 \end{pmatrix}$. Note that the data points in Table 1 are not sorted on x! How does the uncertainty variance σ_m^2 on the slope change?

Problem 3: Write a Metropolis-Hastings sampler for a general one-dimensional probability distribution $p(x)$ with a Gaussian proposal distribution (characterized by a width parameter that should be passed to the code) that returns a sampling and the acceptance fraction. Test it with a Gaussian with zero mean and unit variance: plot a normalized histogram of the samples and compare it to the analytical PDF. Then apply it to sample a probability distribution consisting of the sum of two Gaussians with equal weights, unit variance for each, and means 0 and 10 (again plot a histogram of the samples and the analytical PDF). Try to find a relatively high acceptance fraction.

Problem 4: Solve exercise 6 in HBL using MCMC sampling with `emcee`.


700
600
500
400
300
200
100
0
y
0
50
100
150
200
250
300
x
$y = (2.24 \pm 0.11)\,x + (34 \pm 18)$