Title: Fitting models to data using Markov Chain Monte Carlo

Speakers: Dustin Lang

Collection: Symmetries Graduate School 2023

Date: January 30, 2023 - 2:00 PM

URL: https://pirsa.org/23010089

# emcee: An Affine-Invariant Sampler

Dustin Lang
Perimeter Institute for Theoretical Physics

Symmetries Graduate School 2023-01-30

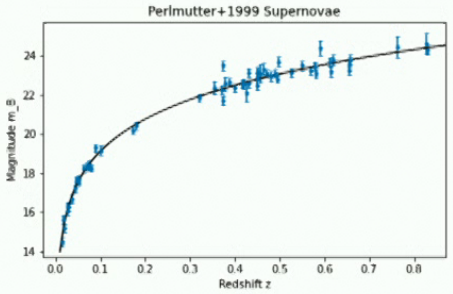
Borrowing heavily from Dan Foreman-Mackey's slides
*https://speakerdeck.com/dfm/data-analysis-with-mcmc1*

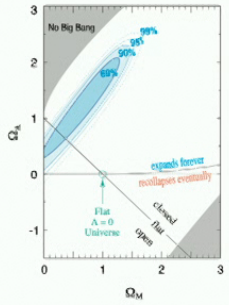
These slides are available at
*https://github.com/dstndstn/MCMC-talk*

# Recap from last week's lecture (1)

- Markov Chain Monte Carlo (MCMC) *draws samples from a probability distribution* when you can *numerically evaluate* the probability function (up to a constant)
- Used extensively in data analysis: *inferring* parameters of models, given observed data
- *Usually* in a Bayesian context; the probability function we run MCMC on is the *posterior* probability:

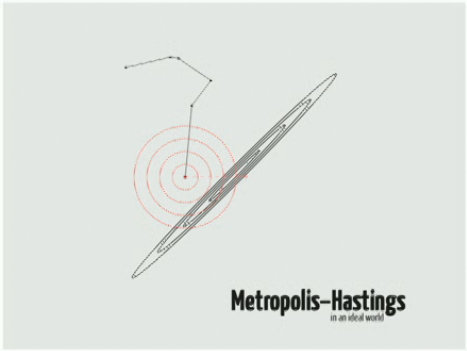
$$\text{posterior}(\text{params}|\text{data}) \propto \text{prior}(\text{params}) \times \text{likelihood}(\text{data}|\text{params})$$



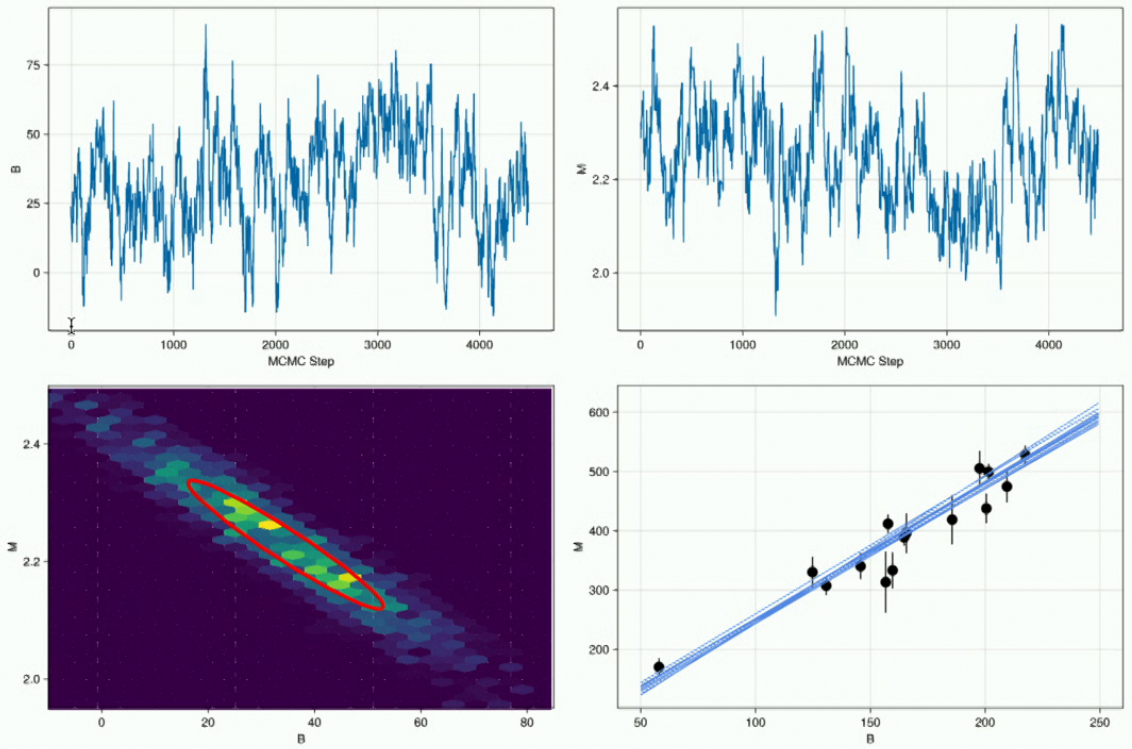



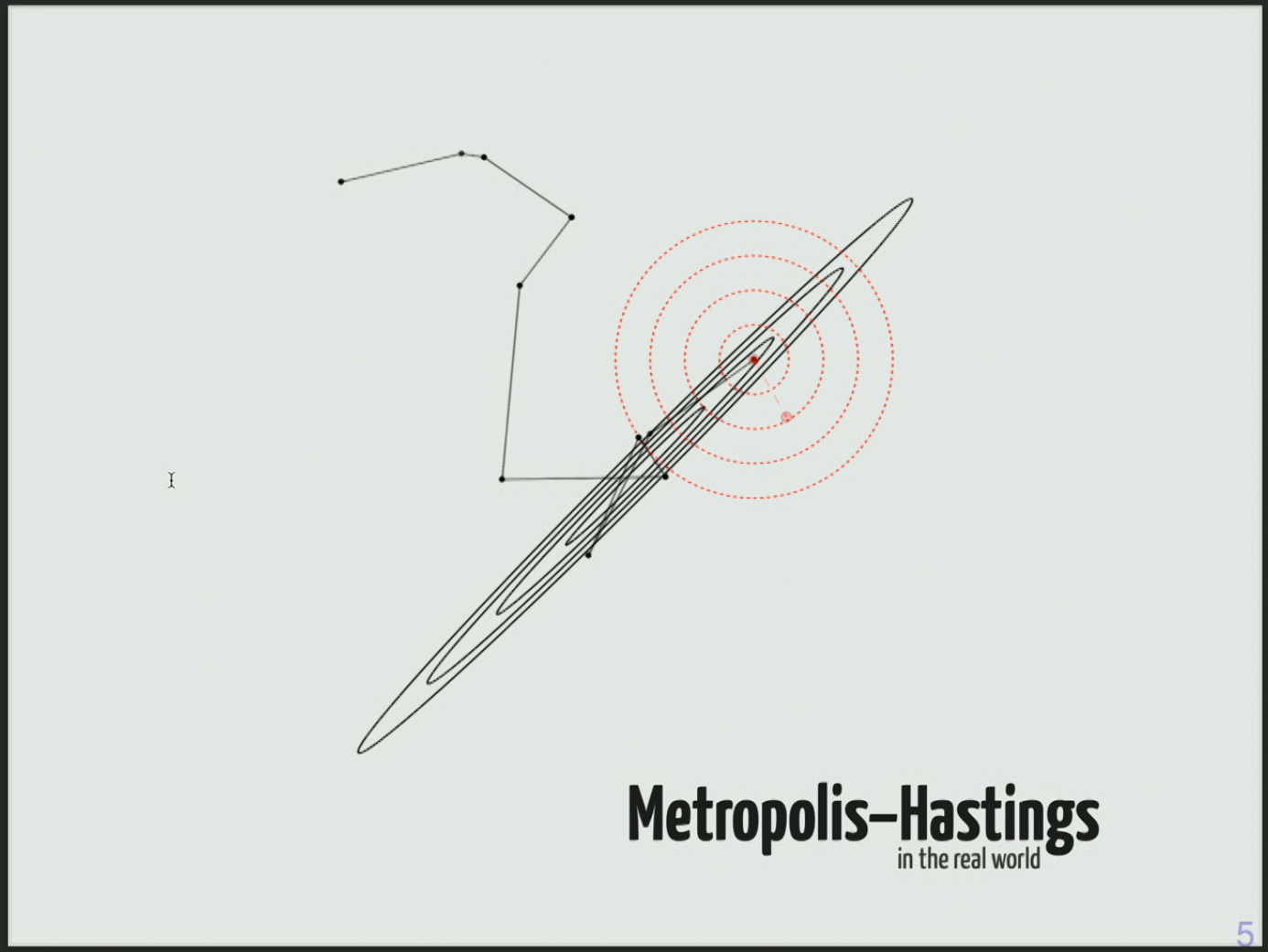
# Recap from last week's lecture (2)

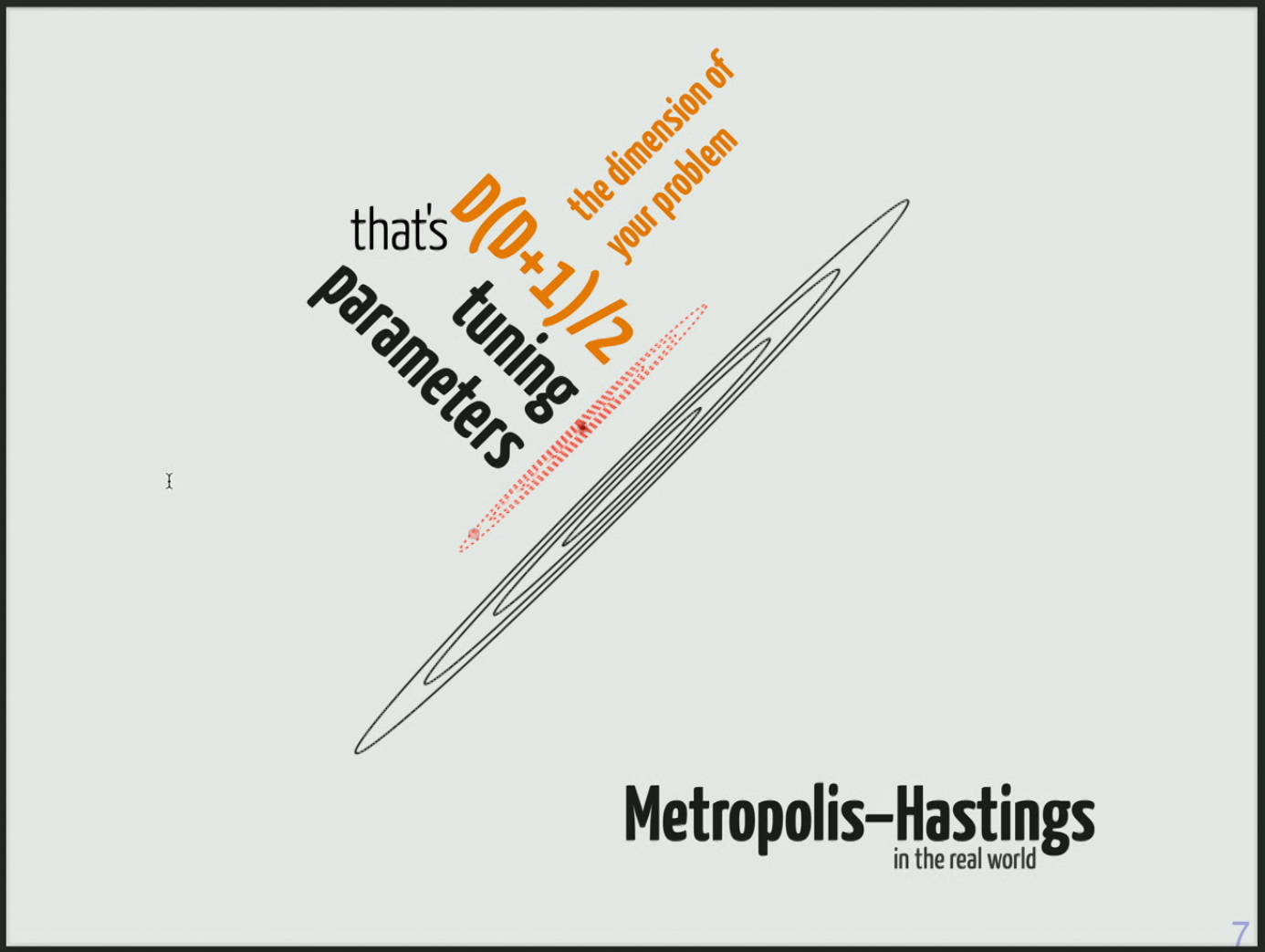
- The "classic" Markov Chain Monte Carlo algorithm is *Metropolis–Hastings*, which moves a *walker* or *particle* around the *state space* (*model parameter space*)
- A randomly-drawn *proposed* jump gets *evaluated* (by calling the probability function), and then *accepted*, or not
- A big difficulty is to *customize* the *proposal distribution* to get the algorithm to work efficiently



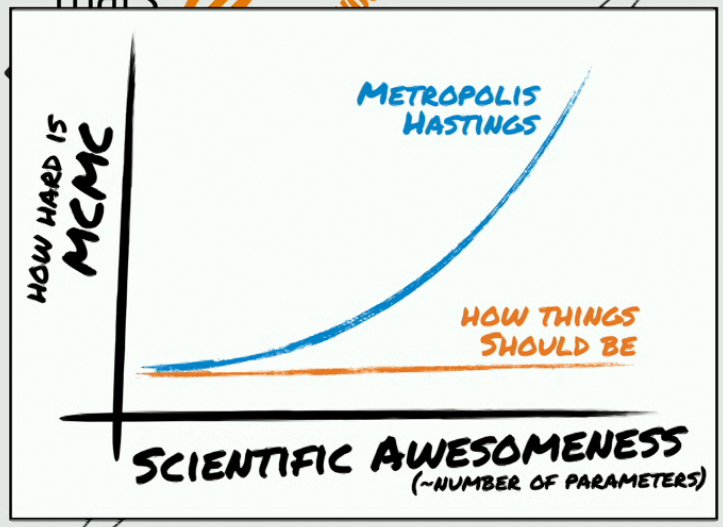
# MCMC for model parameter inference




Metropolis–Hastings
in the real world
5


that's D(D+1)/2 tuning parameters
the dimension of your problem
Metropolis–Hastings
in the real world
7

that's D(D the dimension of your problem




# Metropolis–Hastings
in the real world

Jonathan Goodman

Jonathan Weare

# "Ensemble samplers with affine invariance"

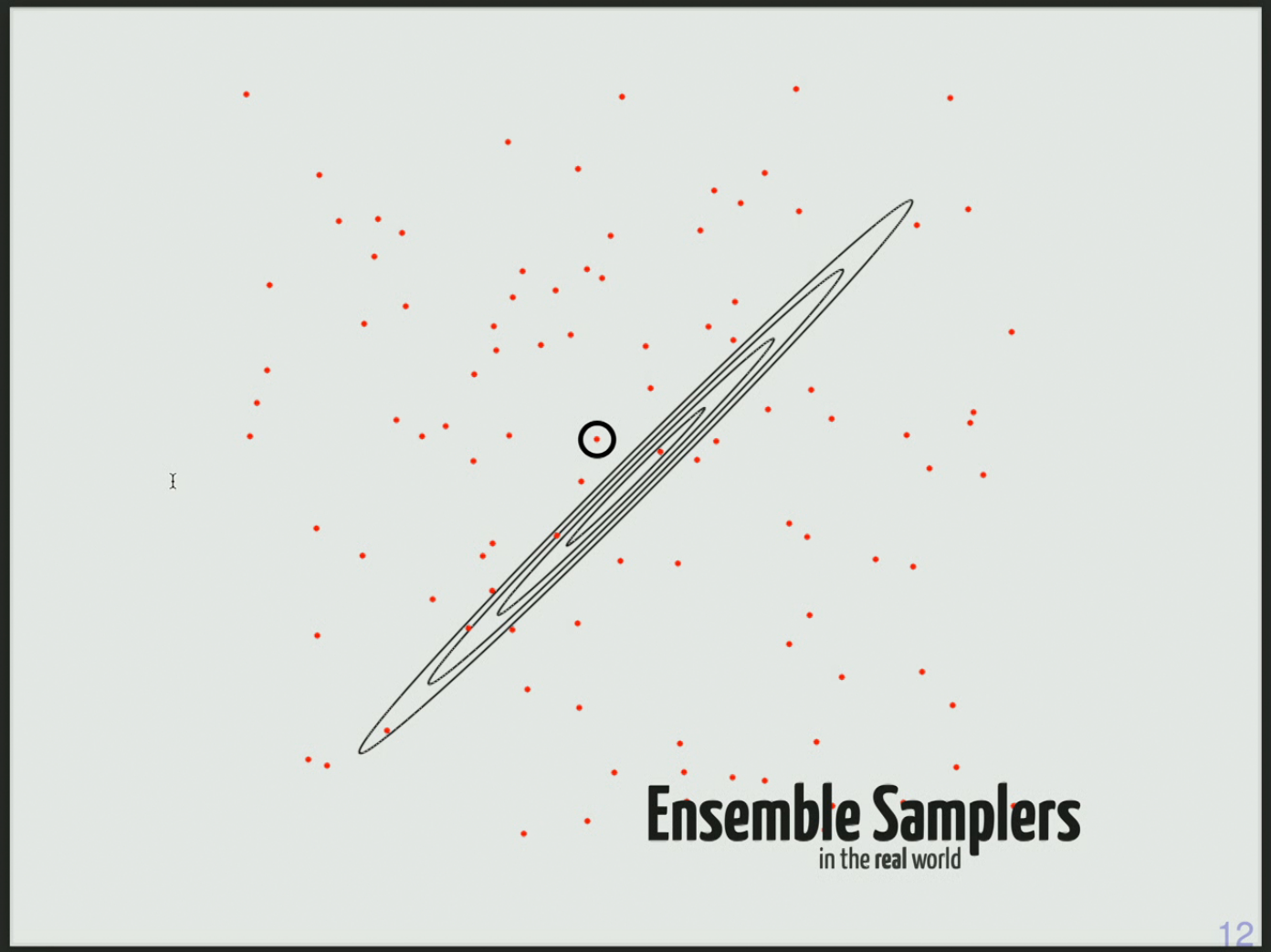
(dfm.io/mcmc-gw10)

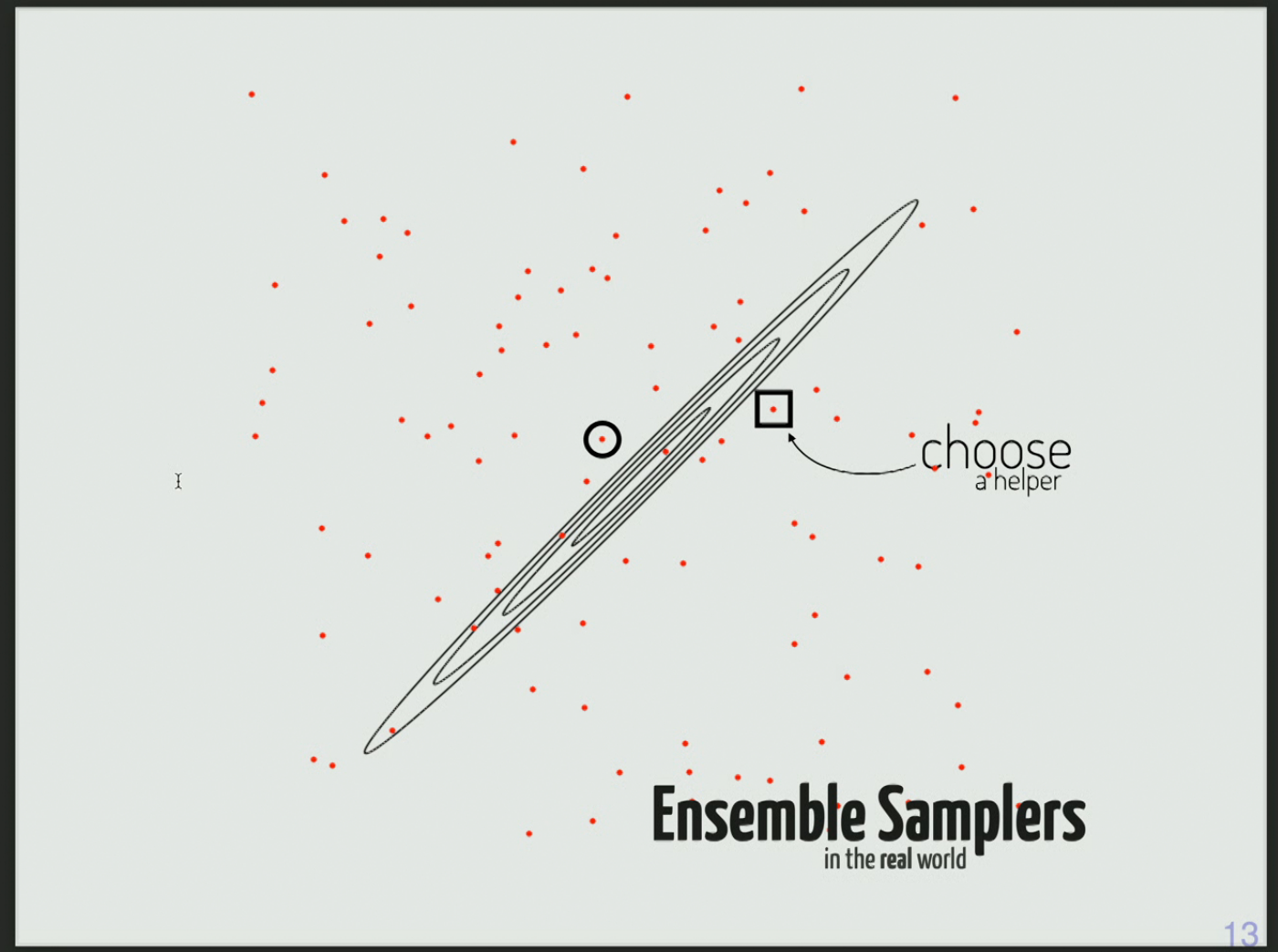

introducing **emcee** the MCMC Hammer

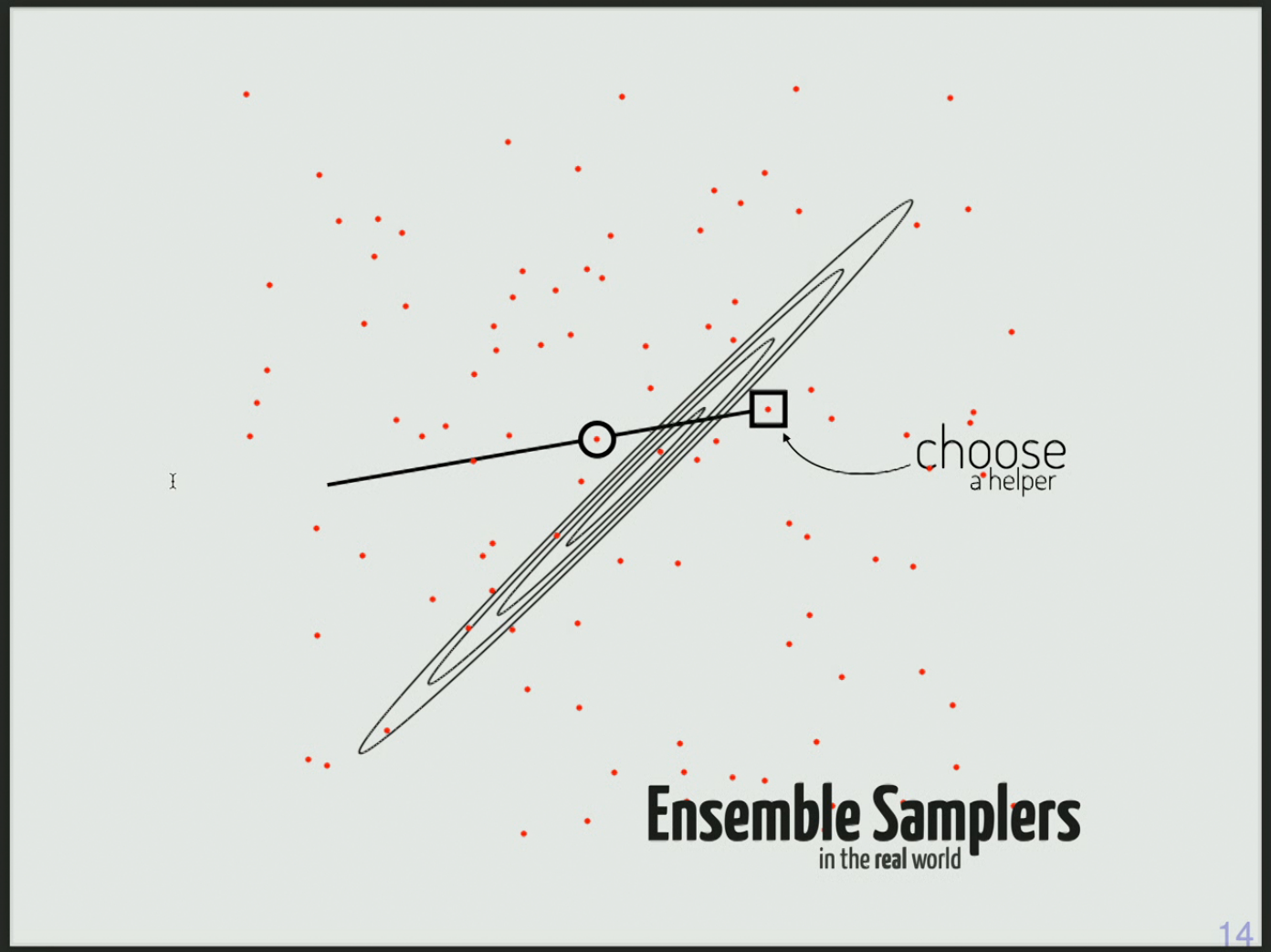
arxiv.org/abs/1202.3665

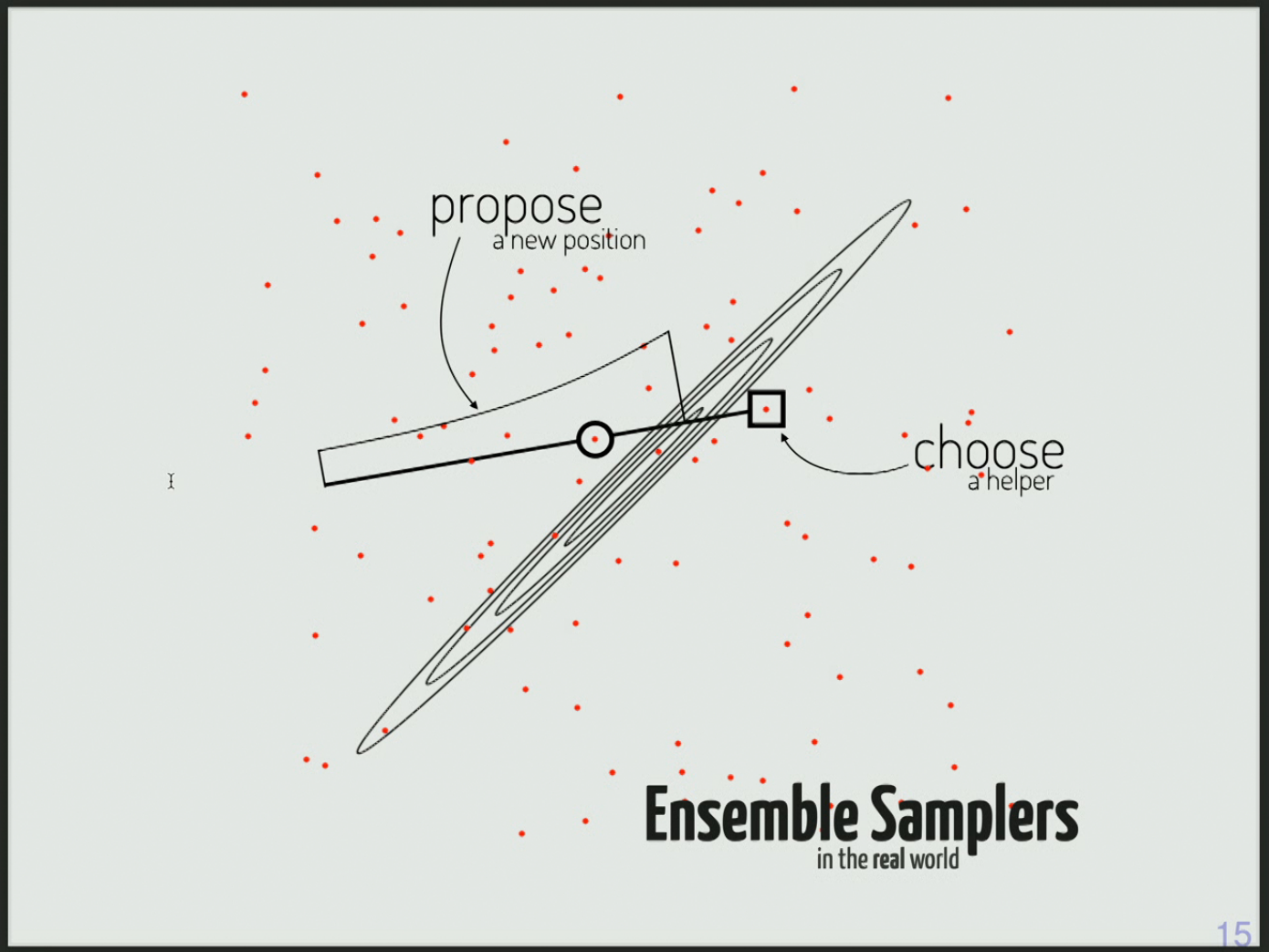
dan.iel.fm/emcee

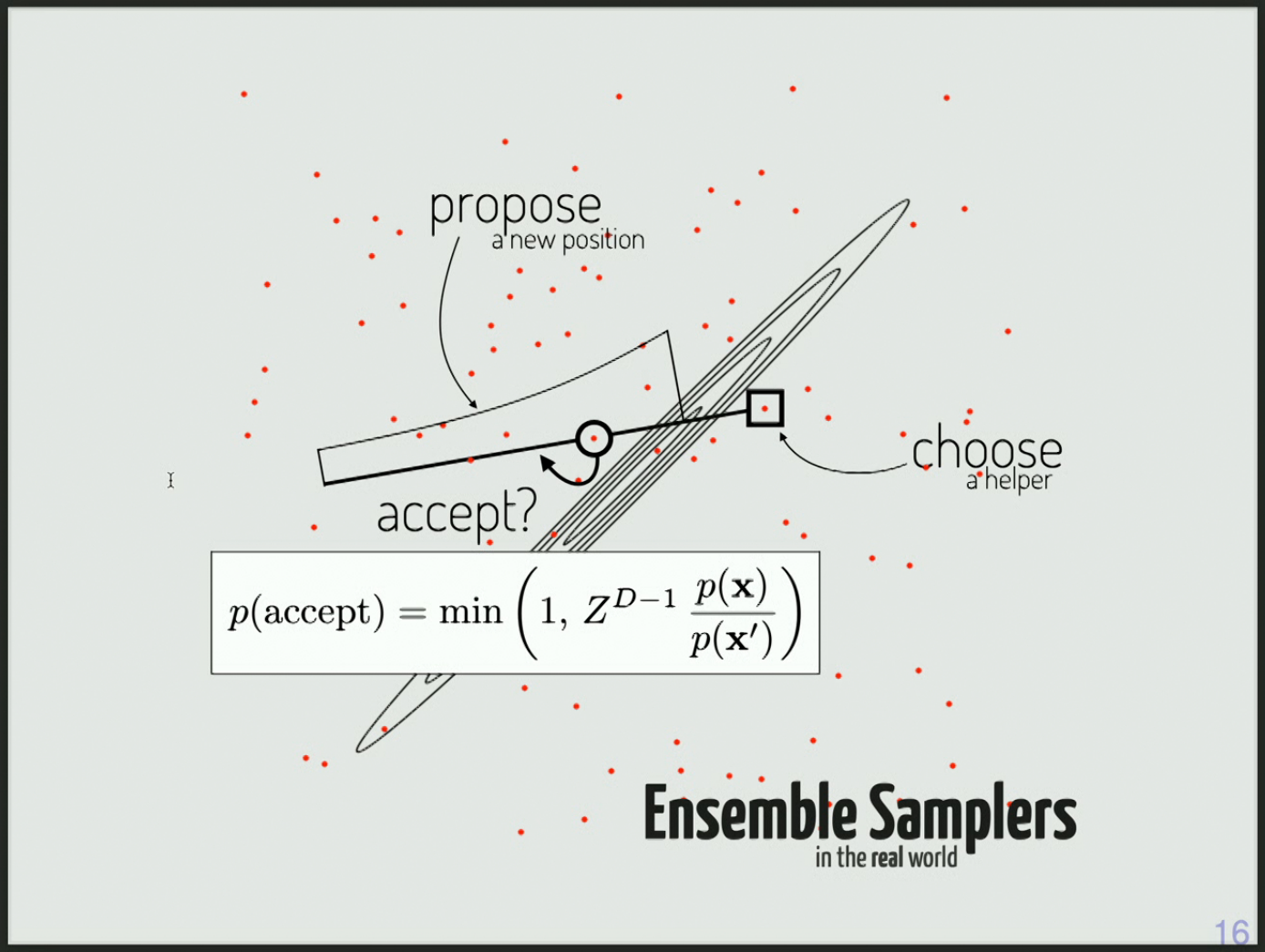

Ensemble Samplers
in the real world
11


Ensemble Samplers
in the real world
12

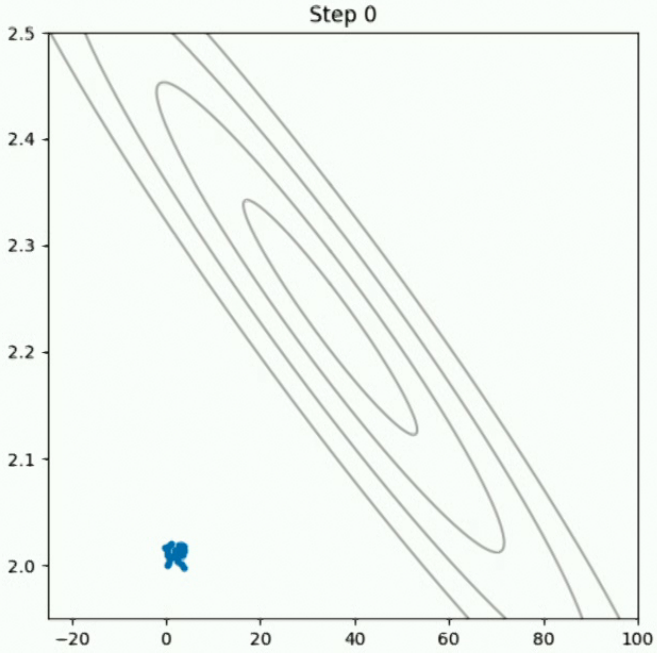

choose
a helper
Ensemble Samplers
in the real world
13


choose
a helper
Ensemble Samplers
in the real world
14

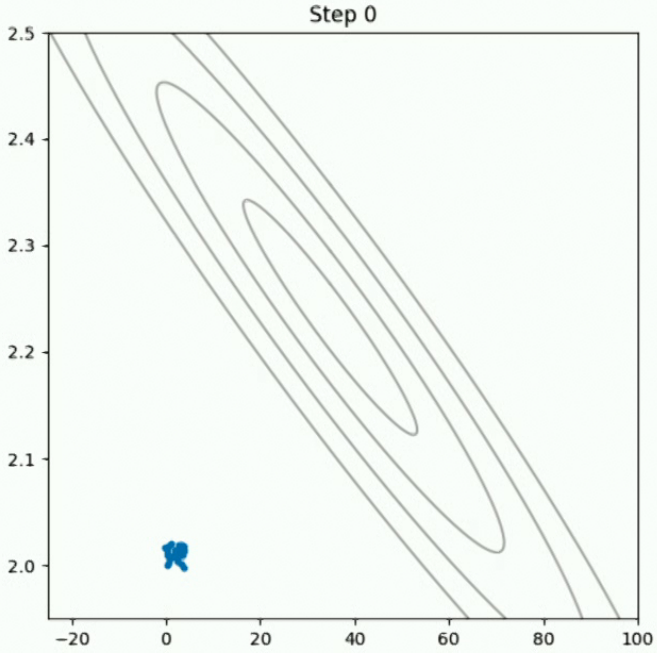

propose
a new position
choose
a helper
Ensemble Samplers
in the real world
15


propose
a new position
choose
a helper
accept?
$p(\text{accept}) = \min\left(1,\, Z^{D-1}\,\frac{p(\mathbf{x})}{p(\mathbf{x}')}\right)$
Ensemble Samplers
in the real world
16

# Emcee demo

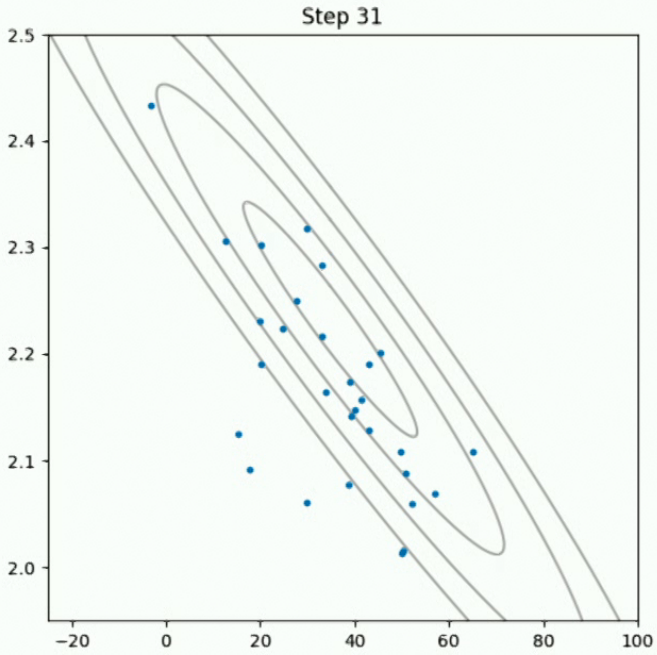


# Emcee demo

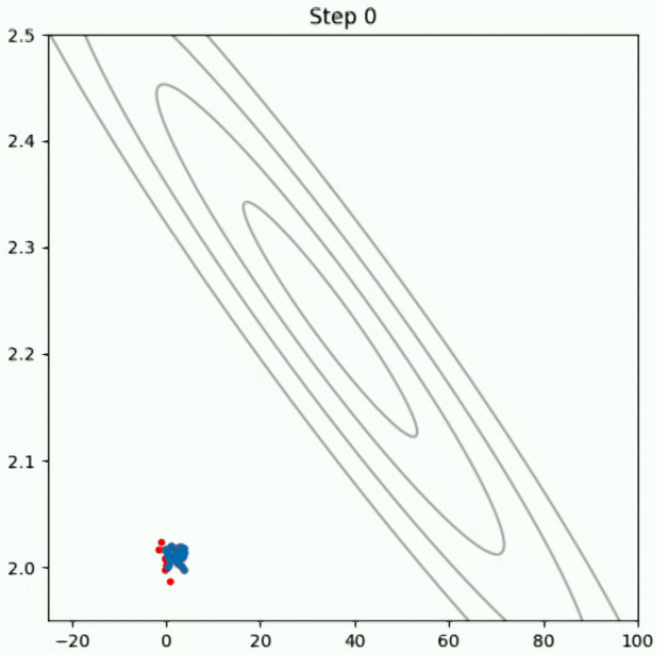



17

# Emcee demo

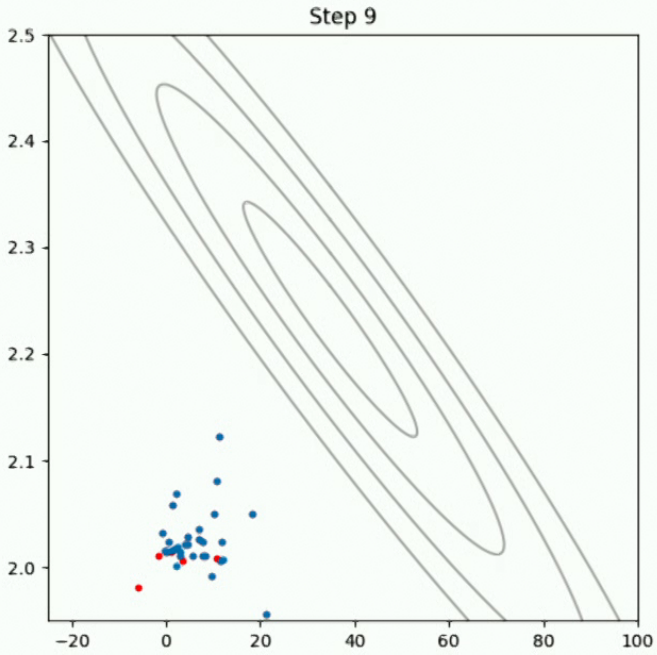


# Emcee demo



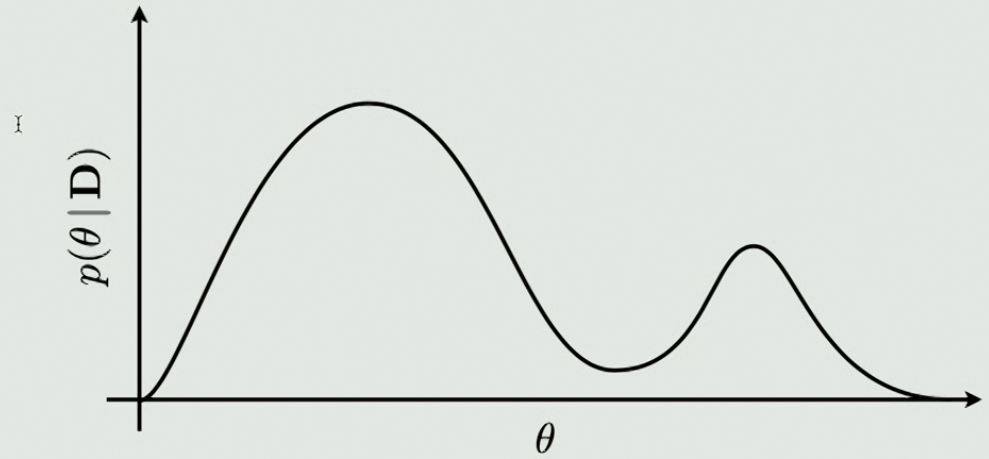

57

# Emcee demo

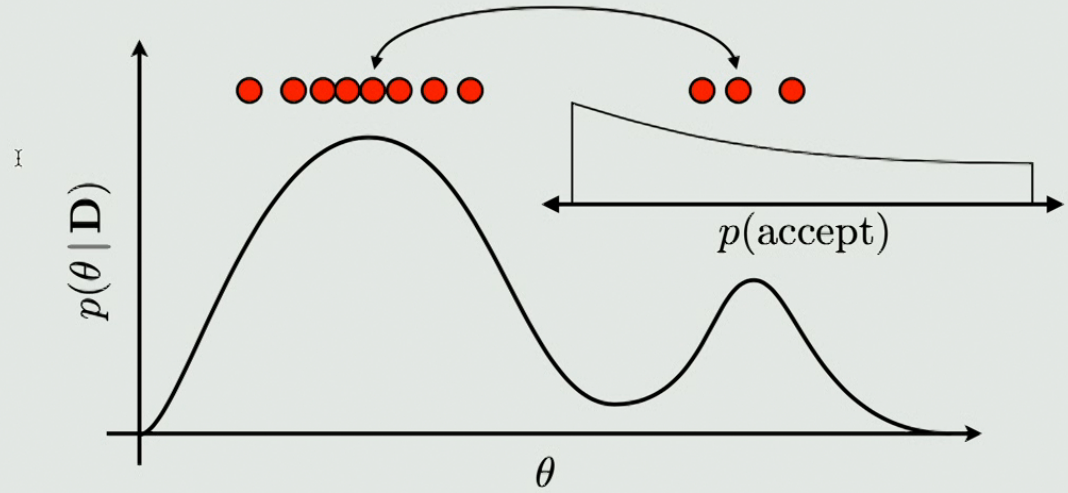



66

# what about multimodal densities?



# what about multimodal densities?



# Differential Evolution move

- emcee allows us to use different *move* types (different *proposal* functions)
- The Differential Evolution (DE) move can improve the sampling for multi-modal distributions
- DE move: randomly select *two* "helpers"
- Propose moving by their vector difference
- (If they are from different modes, this proposes *jumping between modes*)
- Mixing in a fraction of DE moves with the regular "Stretch" move works well!

# Differential Evolution move

- emcee allows us to use different *move* types (different *proposal* functions)
- The Differential Evolution (DE) move can improve the sampling for multi-modal distributions
- DE move: randomly select *two* "helpers"
- Propose moving by their vector difference
- (If they are from different modes, this proposes *jumping between modes*)
- Mixing in a fraction of DE moves with the regular "Stretch" move works well!

# Summary

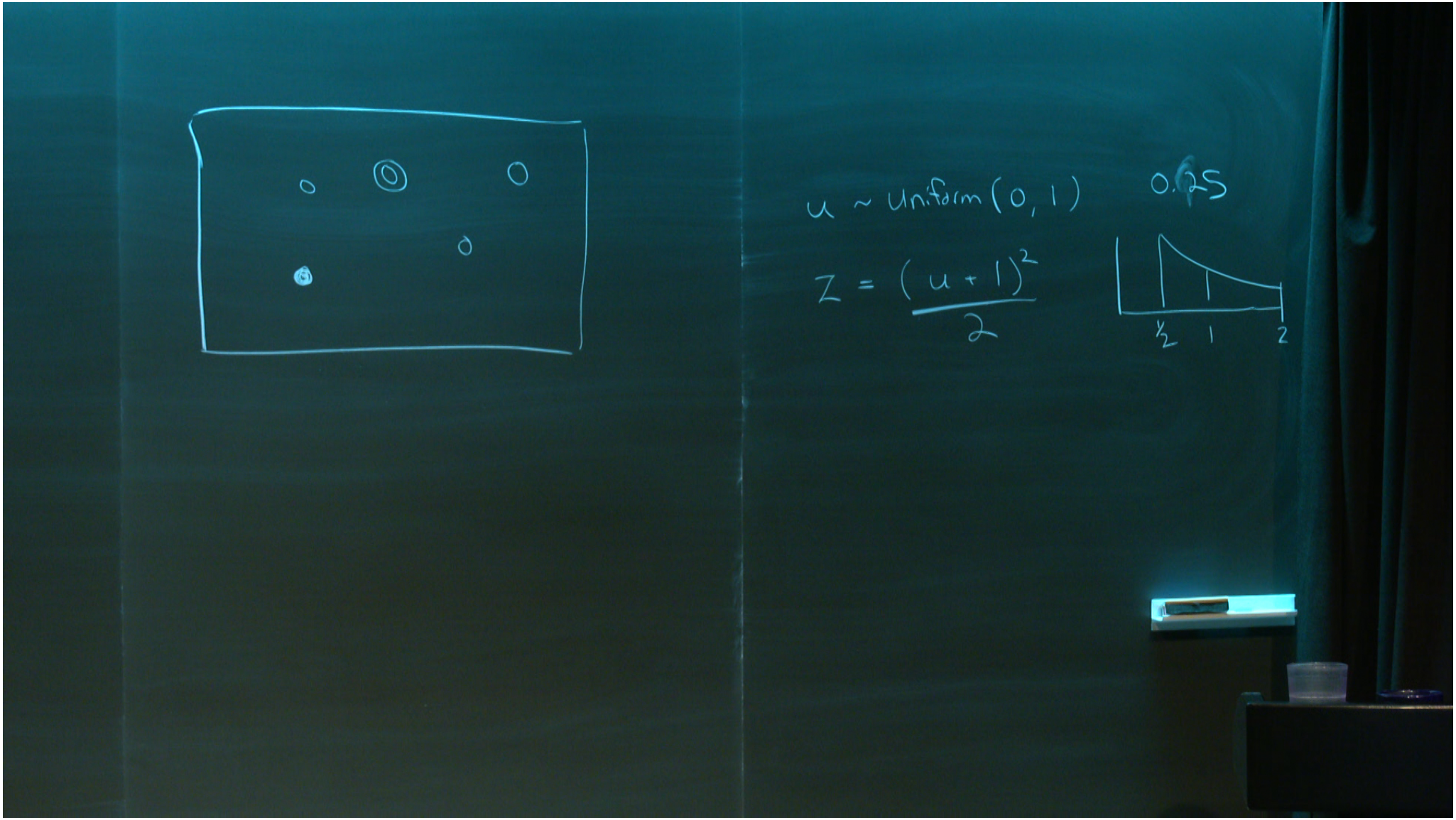
- Traditional Metropolis–Hastings MCMC suffers from a *lack of affine invariance* – requires *tuning parameters* that change for each specific probability function
- *Ensemble samplers* like emcee use the *distribution of the walkers* to achieve *affine invariance*
- $\rightarrow$ much easier to use, and faster sampling
- (Huge side effect: parallelizable!)
- Multi-modal distributions still hard, but *DE Move* can help
- MCMC isn't scary!

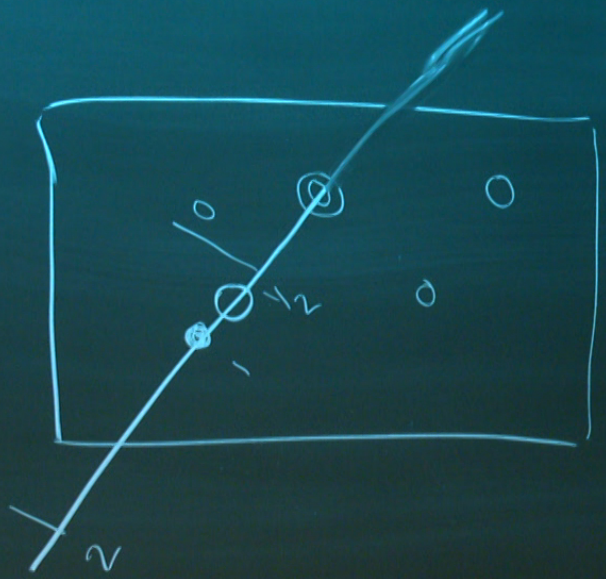
**Algorithm 2** A single stretch move update step from GW10

1: **for** $k = 1, \ldots, K$ **do**
2: Draw a walker $X_j$ at random from the complementary ensemble $S_{[k]}(t)$
3: $z \leftarrow Z \sim g(z)$, Equation (10)
4: $Y \leftarrow X_j + z\,[X_k(t) - X_j]$
5: $q \leftarrow z^{N-1}\,p(Y)/p(X_k(t))$ *// This line is generally expensive*
6: $r \leftarrow R \sim [0, 1]$
7: **if** $r \leq q$, Equation (9) **then**
8: $X_k(t+1) \leftarrow Y$
9: **else**
10: $X_k(t+1) \leftarrow X_k(t)$
11: **end if**
12: **end for**

**The parallel stretch move** It is tempting to parallelize the stretch move algorithm by simultaneously advancing each walker based on the state of the ensemble instead of evolving the walkers in series. Unfortunately, this subtly violates detailed balance. Instead, we must split the full ensemble into two subsets ($S^{(0)} = \{X_k, \forall k = 1, \ldots, K/2\}$ and $S^{(1)} = \{X_k, \forall k = K/2+1, \ldots, K\}$) and simultaneously update all the walkers in $S^{(0)}$ — using the stretch move procedure from Algorithm 2 — based *only* on the positions of the walkers in the other set ($S^{(1)}$). Then, using the new positions $S^{(0)}$, we can update $S^{(1)}$. In this case, the outcome is a valid step for all of the walkers. The pseudocode for this procedure is shown in Algorithm 3. This code is similar to Algorithm 2 but now the computationally expensive inner loop (starting at line 2 in Algorithm 3) can be run in parallel.

The performance of this method — quantified by the autocorrelation time — is compa-


$u \sim \text{Uniform}(0,1)$
0.25
$Z = \frac{(u+1)^2}{2}$
½
1
2




$u \sim \text{Uniform}(0,1)$ 0.25

$Z = \frac{(u+1)^2}{2}$