Probabilistic
Graphical
Models

Inference

Sampling Methods

Using a Markov Chain

Using a Markov Chain

- Goal: compute $P(x \in S)$
  - but P is too hard to sample from directly
- Construct a regular Markov chain T whose unique stationary distribution is P
- Sample $x^{(0)}$ from some $P^{(0)}$
- For t = 0, 1, 2, …
  - Generate $x^{(t+1)}$ from $T(x^{(t)} \to x')$

Using a Markov Chain

- We only want to use samples that are sampled from a distribution close to P
- At early iterations, $P^{(t)}$ is usually far from P
- Start collecting samples only after the chain has run long enough to "mix" $P^{(t)}$ close enough to π

Daphne Koller

Mixing

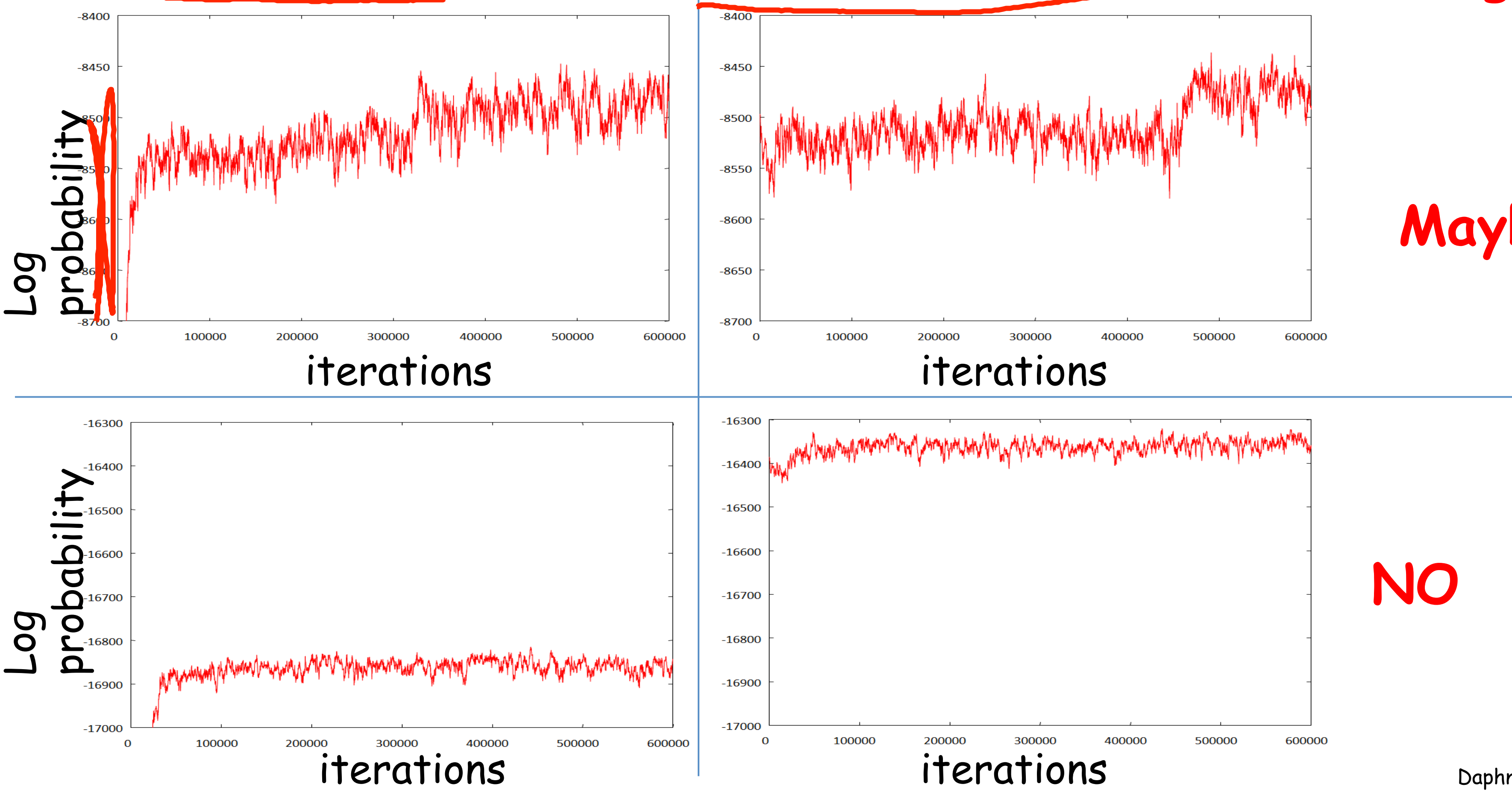
- How do you know if a chain has mixed or not?
  - In general, you can never "prove" a chain has mixed
  - But in many cases you can show that it has NOT

- How do you know a chain has not mixed?
  - Compare chain statistics in different windows within a single run of the chain
  - and across different runs initialized differently

Mixing?

Initialized from an arbitrary state

Initialized from a high-probability state

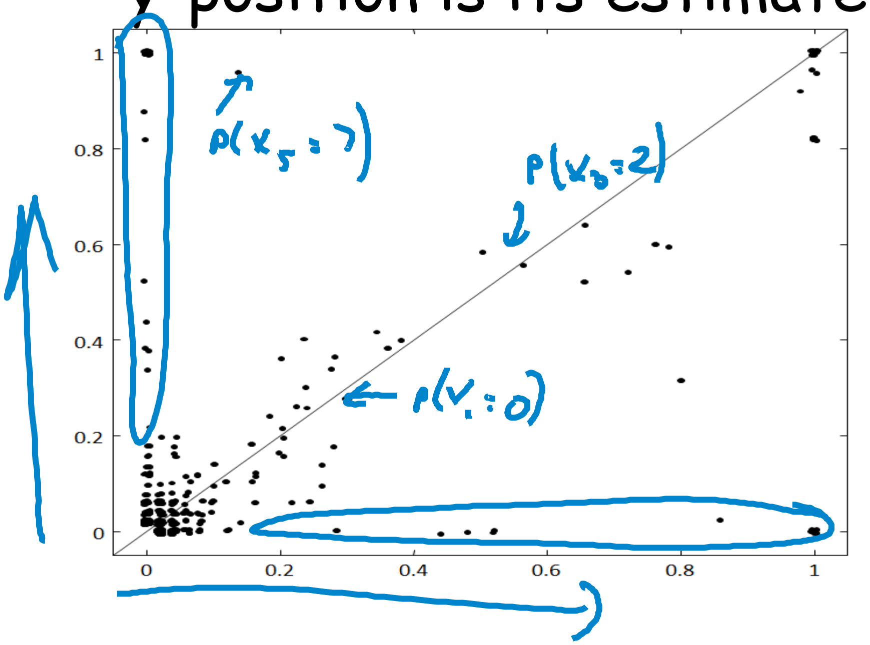


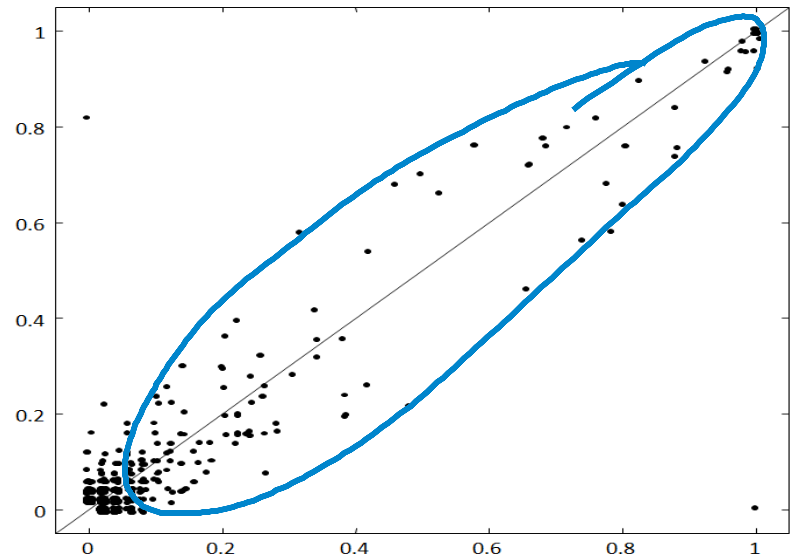
Maybe

NO

Daphne Koller

- Each dot is a statistic (e.g., $P(x \in S)$)
- x-position is its estimated value from chain 1
- y-position is its estimated value from chain 2





Mixing? **NO** **Maybe**

Daphne Koller

Using the Samples

- Once the chain mixes, all samples $x^{(t)}$ are from the stationary distribution π
  - So we can (and should) use all $x^{(t)}$ for $t > T_{mix}$
- However, nearby samples are correlated!
  - So we shouldn't overestimate the quality of our estimate by simply counting samples   not IID
- The faster a chain mixes, the less correlated (more useful) the samples

MCMC Algorithm Summary I

- For c=1,…,C
  - Sample $x^{(c,0)}$ from $P^{(0)}$
- Repeat until mixing
  - For c=1,…,C
    - Generate $x^{(c,\,t+1)}$ from $T(x^{(c,t)} \rightarrow x')$
  - Compare window statistics in different chains to determine mixing
  - t := t+1

MCMC Algorithm Summary II

- Repeat until sufficient samples
  - D := ∅
  - For c=1,…,C
    - Generate $x^{(c,\,t+1)}$ from $T(x^{(c,t)} \rightarrow x')$
    - $D := D \cup \{x^{(c,\,t+1)}\}$
  - t := t+1
- Let D = {x[1],…,x[M]}
- Estimate $$\boldsymbol{E}_P[f] \approx \frac{1}{M}\sum_{m=1}^{M} f(x[m])$$

Summary

- Pros:
  - Very general purpose
  - Often easy to implement
  - Good theoretical guarantees as $t \to \infty$
- Cons:
  - Lots of tunable parameters / design choices
  - Can be quite slow to converge
  - Difficult to tell whether it's working