



# Markov Chains + Bayesian Inference for Parameter Estimation

Matt Gormley  
Lecture 13  
Oct. 12, 2022

# Reminders

- **Homework 2: Learning to Search for RNNs**
  - **Programming + Empirical Questions**
    - **Due: Mon, Oct 24 at 9:00am**
  - **Policy: 65 points or more on the autograder gives 100% autograder credit**
- **Homework 3: General Graph CRF Module**
  - **Out: Thu, Sep 29**
  - **Due: Mon, Oct 10 at 11:59pm**
- **Practice Problems 1**
- **Exam 1: Fri, Oct 14, in-class**

# **METROPOLIS-HASTINGS**

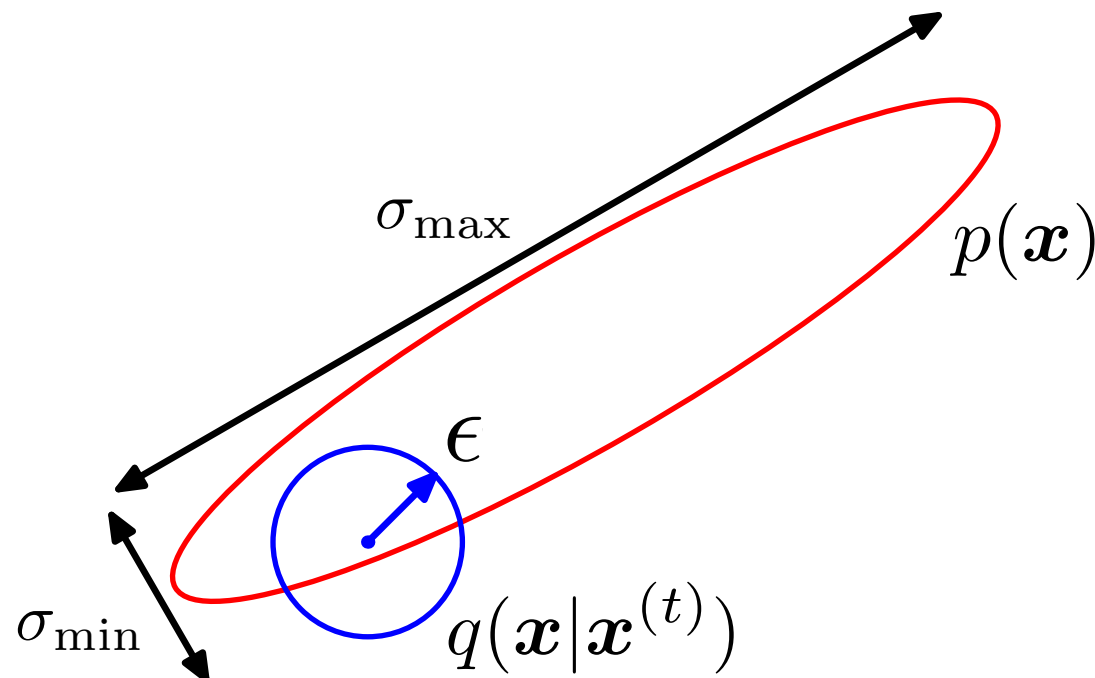
# Metropolis-Hastings

## ***Whiteboard***

- Metropolis Algorithm
- Metropolis-Hastings Algorithm

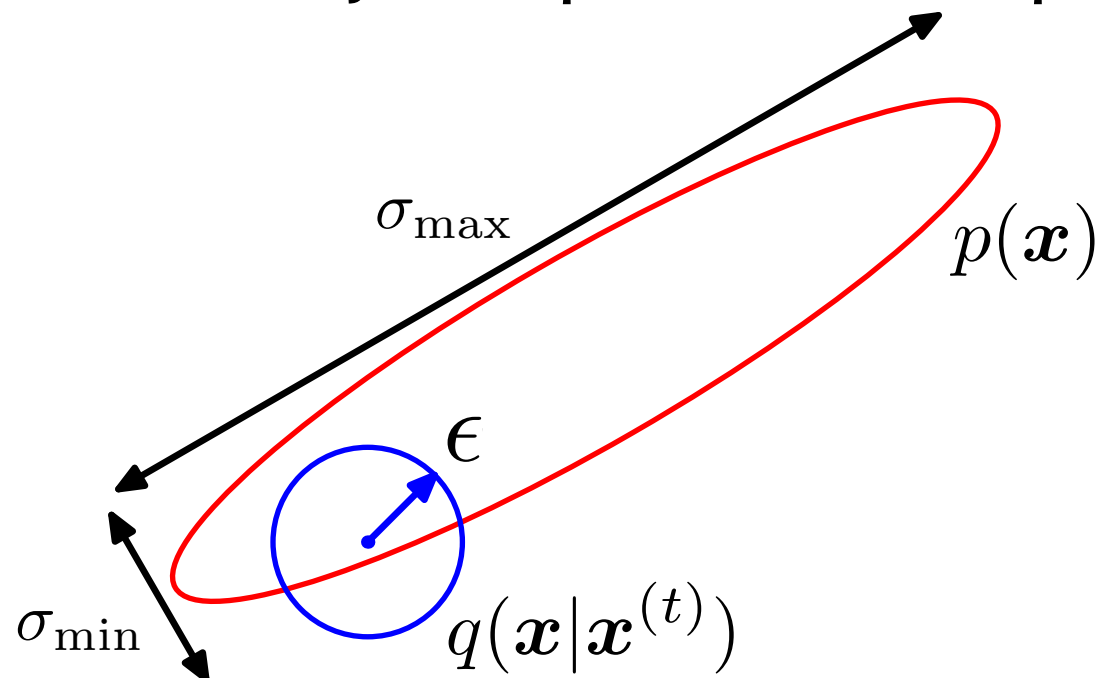
# Random Walk Behavior of M-H

- For **Metropolis-Hastings**, a generic proposal distribution is:  $q(x|x^{(t)}) = \mathcal{N}(0, \epsilon^2)$
- If  $\epsilon$  is large, many rejections
- If  $\epsilon$  is small, slow mixing



# Random Walk Behavior of M-H

- For **Rejection Sampling**, the accepted samples are **independent**
- But for **Metropolis-Hastings**, the samples are **correlated**
- **Question:** How long must we wait to get effectively independent samples?



**A:** independent states in the M-H random walk are separated by roughly  $(\sigma_{\max}/\sigma_{\min})^2$  steps

# Gibbs Sampling as M-H

- Gibbs Sampling is a special case of Metropolis-Hastings

Let  $q(\vec{x}|\vec{x}^{(t)}) \triangleq \begin{cases} x_i \sim p(x_i | \vec{x}_{-i}^{(t)}) & \text{where } i \sim \text{Unif}(1, \dots, J) \\ \vec{x}_{-i} = \vec{x}_{-i}^{(t)} \end{cases}$

$$A(\vec{x} \leftarrow \vec{x}^{(t)}) = \min \left( 1, \frac{\hat{p}(\vec{x})}{\hat{p}(\vec{x}^{(t)})} \frac{q(\vec{x}^{(t)}|\vec{x})}{q(\vec{x}|\vec{x}^{(t)})} \right)$$

$$= \min \left( 1, \frac{p(\vec{x})}{p(\vec{x}^{(t)})} \frac{p(x_i^{(t)}|\vec{x}_{-i})}{p(x_i|\vec{x}_{-i}^{(t)})} \right)$$

$$p(a,b) = p(a|b)p(b)$$

$$= \min \left( 1, \frac{p(x_i|\vec{x}_{-i})p(\vec{x}_{-i})}{p(x_i^{(t)}|\vec{x}_{-i}^{(t)})p(\vec{x}_{-i}^{(t)})} \frac{p(x_i^{(t)}|\vec{x}_{-i})}{p(x_i|\vec{x}_{-i}^{(t)})} \right)$$

$\vec{x}_{-i} = \vec{x}_{-i}^{(t)}$ 
 $x_i \neq x_i^{(t)}$

$$= \min(1, 1)$$

# **MCMC PRACTICAL ISSUES**



# Practical Issues

- **Question:** Is it better to move along one dimension or many?
- **Answer:** For **Metropolis-Hastings**, it is sometimes better to sample one dimension at a time
  - Q: Given a sequence of 1D proposals, compare rate of movement for **one-at-a-time** vs. **concatenation**.
- **Answer:** For **Gibbs Sampling**, sometimes better to sample a block of variables at a time
  - Q: When is it tractable to sample a block of variables?

# Blocked Gibbs Sampling

## Goal:

Draw samples from a distribution  $y_1, y_2, \dots, y_J \sim p(y_1, y_2, \dots, y_J)$

## Algorithm:

- Initialize  $y_1, y_2, \dots, y_J$  to arbitrary values
- For  $t = 1, 2, \dots$ :
  - for  $b$  in  $B$ :
    - where  $b \subseteq \{1, \dots, J\}$
    - $y_b \sim p(y_b \mid y_{-b})$
- Example:  $B$  = set of factors in a factor graph

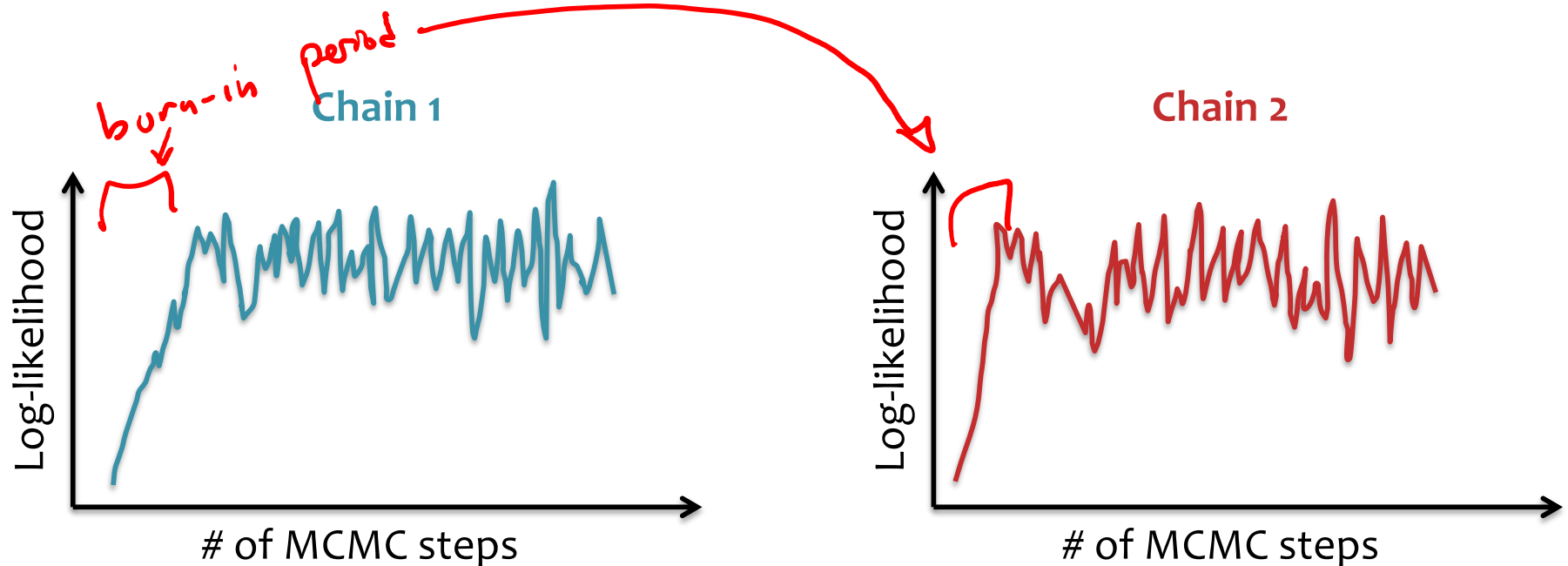
e.g.  $b = \{7, 10, 13\}$   
 $y_b = \{y_7, y_{10}, y_{13}\}$

## Why use blocks?

- As in Gibbs Sampler, this will eventually yield samples from  $p(y_1, y_2, \dots, y_J)$
- **Might improve mixing time** (i.e. “eventually” will be a bit sooner)

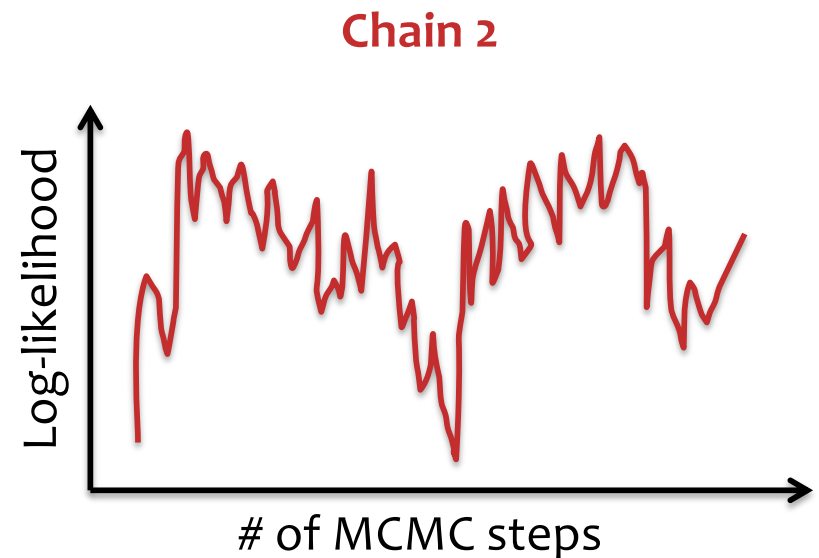
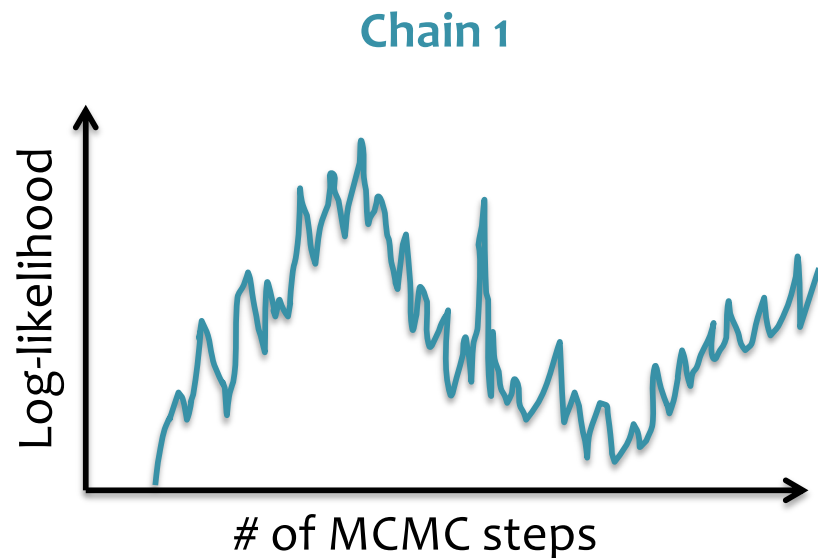
# Practical Issues

- **Question:** How do we assess convergence of the Markov chain?
- **Answer:** It's not easy!
  - Compare statistics of multiple independent chains
  - Ex: Compare log-likelihoods



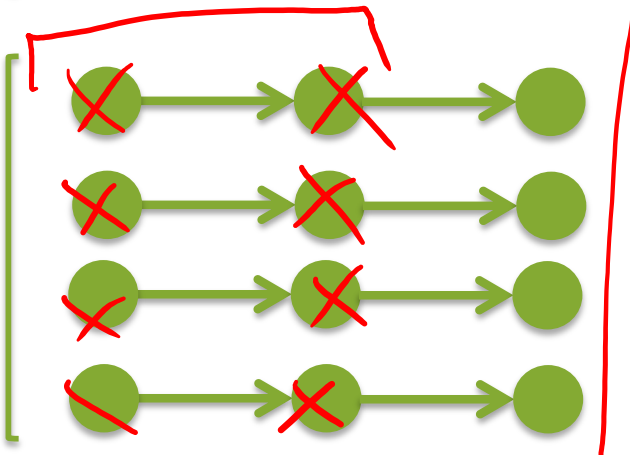
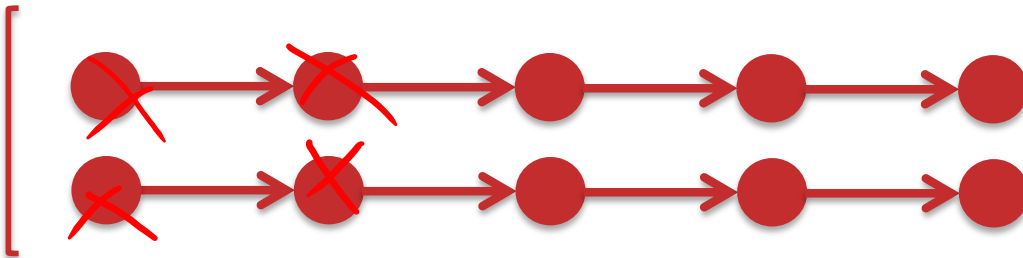
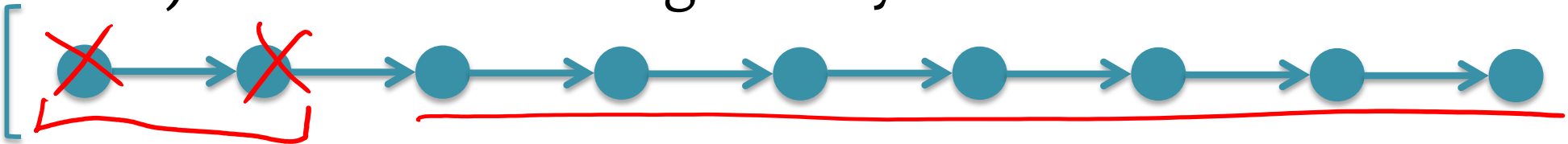
# Practical Issues

- **Question:** How do we assess convergence of the Markov chain?
- **Answer:** It's not easy!
  - Compare statistics of multiple independent chains
  - Ex: Compare log-likelihoods



# Practical Issues

- **Question:** Is one long Markov chain better than many short ones?
- **Note:** typical to discard initial samples (aka. “burn-in”) since the chain might not yet have mixed



- **Answer:** Often a balance is best:
  - Compared to one long chain: More independent samples
  - Compared to many small chains: Less samples discarded for burn-in
  - We can still parallelize
  - Allows us to assess mixing by comparing chains

# MCMC Summary

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

Definitions and Theoretical Justification for MCMC


# **MARKOV CHAINS**

# Markov Chains

- a **Markov chain** is a random process  
     $\hookrightarrow$  gives a series of random variables

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}, \mathbf{x}^{(t+1)}$$

- **first order Markov chain:**

$$p(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) = p(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)})$$


we're focused  
on first order  
only

- **second order Markov chain:**

$$p(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) = p(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)})$$

- **transition probabilities:**

$$R_t(\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)}) \triangleq p(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)})$$

- **homogeneous Markov chain:**  $R_t \triangleq R$ , i.e. the transition probabilities are the same for all  $t$



# Markov Chains

## *Whiteboard*

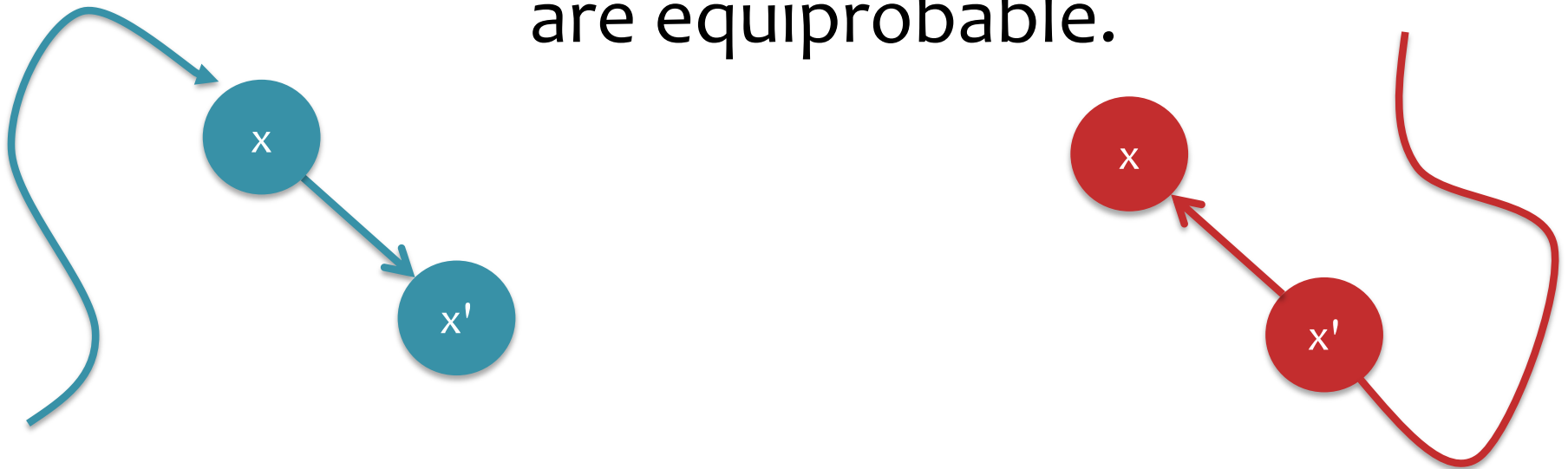
- Invariant distribution
- Equilibrium distribution
- Sufficient conditions for MCMC
- Markov chain as a WFSM

# Detailed Balance

$$S(x' \leftarrow x)p(x) = S(x \leftarrow x')p(x')$$

Detailed balance means that, for each pair of states  $x$  and  $x'$ ,

arriving at  $x$  then  $x'$  and arriving at  $x'$  then  $x$  are equiprobable.



# MCMC Summary

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

Slice Sampling, Hamiltonian Monte Carlo

# **MCMC (AUXILIARY VARIABLE METHODS)**

# Auxiliary variables

The point of MCMC is to marginalize out variables, but one can introduce more variables:

$$\begin{aligned}\int f(x)P(x) \, dx &= \int f(x)P(x, v) \, dx \, dv \\ &\approx \frac{1}{S} \sum_{s=1}^S f(x^{(s)}), \quad x, v \sim P(x, v)\end{aligned}$$

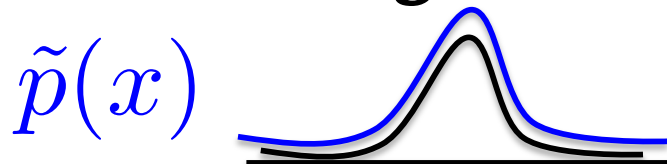
**We might want to do this if**

- $P(x|v)$  and  $P(v|x)$  are simple
- $P(x, v)$  is otherwise easier to navigate

# Slice Sampling

Extra Slides

- Motivation:
  - Want **samples** from  $p(x)$  and don't know the normalizer  $Z$
  - Choosing a proposal at the correct **scale** is difficult
- Properties:
  - Similar to *Gibbs Sampling*: **one-dimensional** transitions in the state space
  - Similar to *Rejection Sampling*: (asymptotically) draws samples from the **region under the curve**

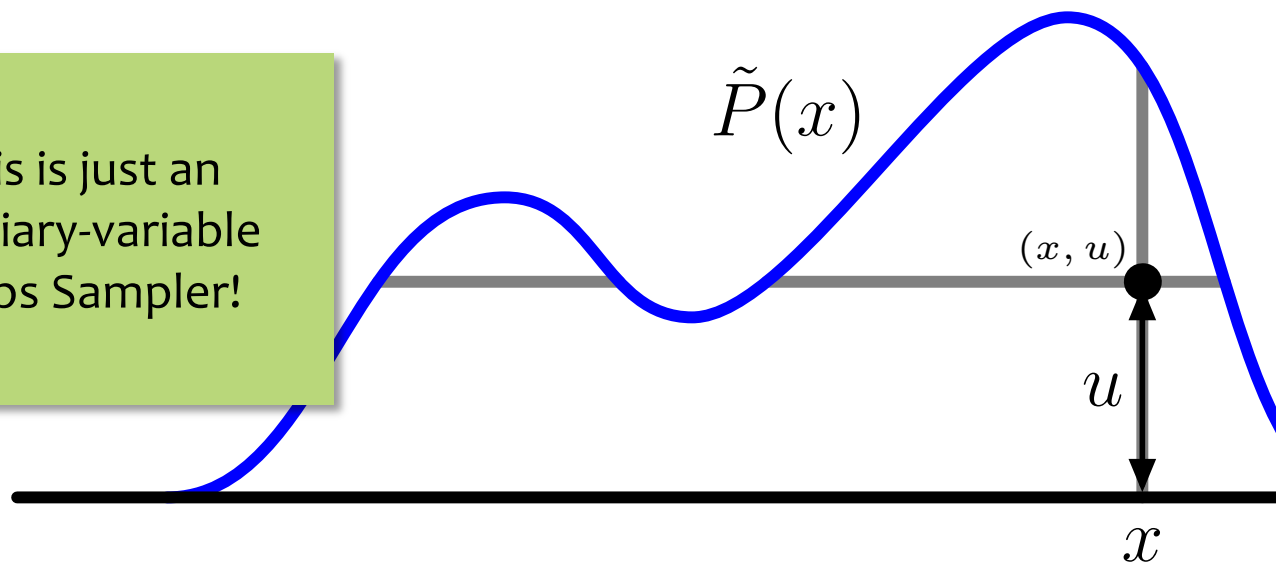


- An MCMC method with an **adaptive proposal**

# Slice sampling idea

Sample point uniformly under curve  $\tilde{P}(x) \propto P(x)$

This is just an  
auxiliary-variable  
Gibbs Sampler!

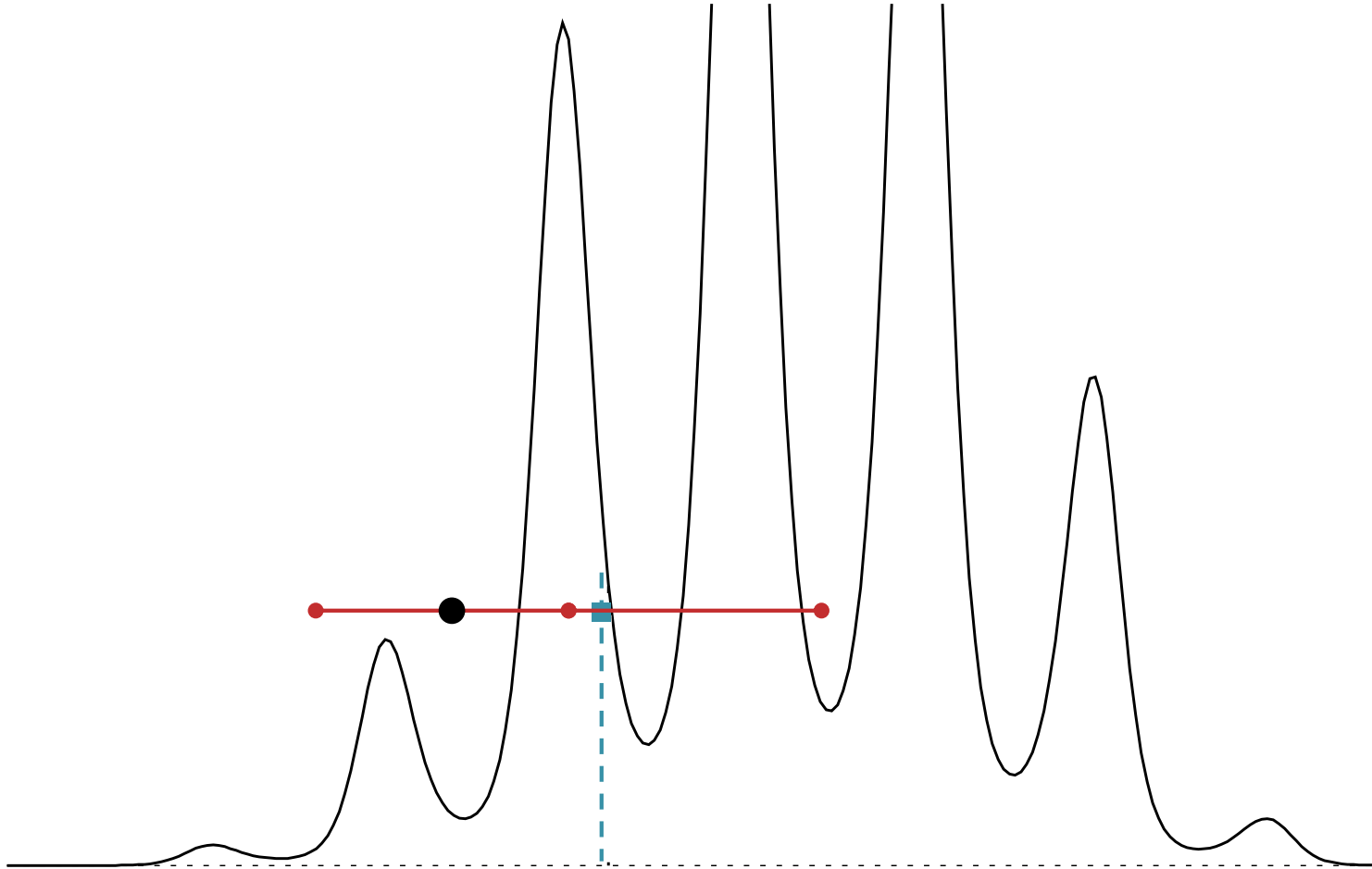


Problem: Sampling  
from the conditional  
 $p(x | u)$  might be  
infeasible.

$$p(u|x) = \text{Uniform}[0, \tilde{P}(x)]$$

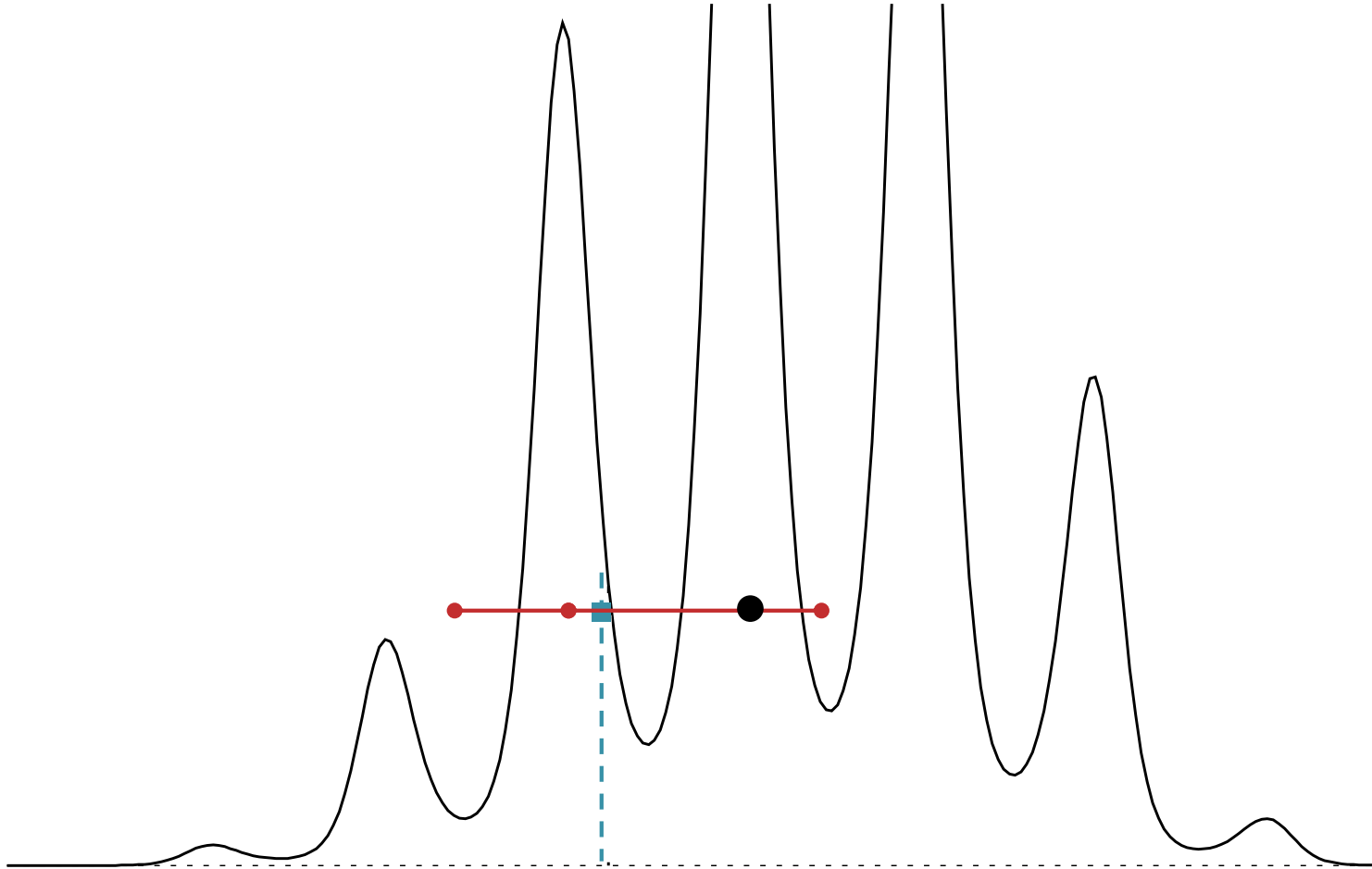
$$p(x|u) \propto \begin{cases} 1 & \tilde{P}(x) \geq u \\ 0 & \text{otherwise} \end{cases} = \text{"Uniform on the slice"}$$

# Slice Sampling

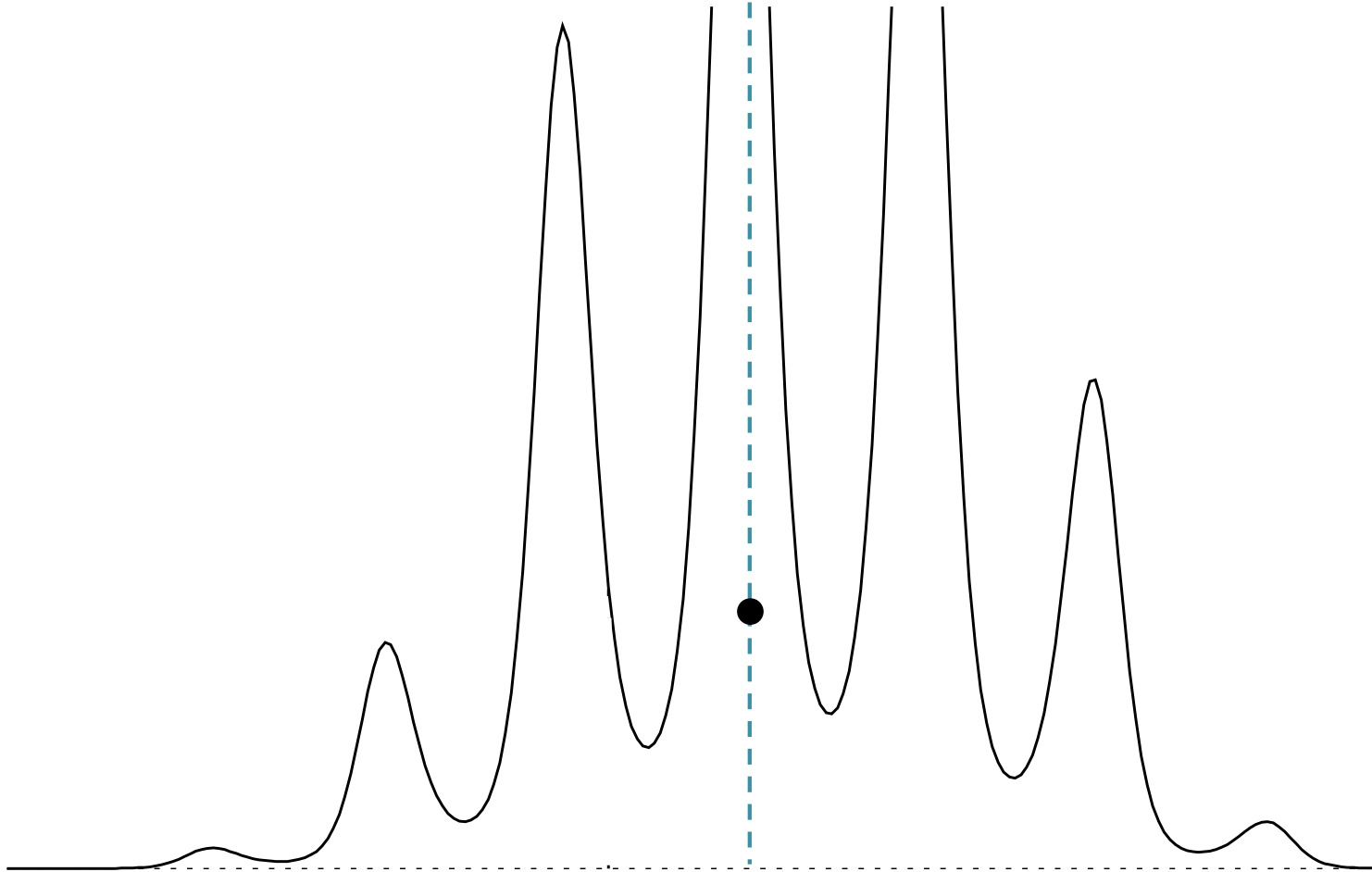




# Slice Sampling



# Slice Sampling



# Slice Sampling

Extra Slides

**Goal:** sample  $(x, u)$  given  $(u^{(t)}, x^{(t)})$ .

**Part 1:** Stepping Out

Sample interval  $(x_l, x_r)$  enclosing  $x^{(t)}$ .

Expand until endpoints are "outside" region under curve.

**Part 2:** Sample  $x$  (Shrinking)

Draw  $x$  from within the interval  $(x_l, x_r)$ , then accept or shrink.

Algorithm:

# Slice Sampling

Extra Slides

Algorithm:

**Goal:** sample  $(x, u)$  given  $(u^{(t)}, x^{(t)})$ .

$u \sim \text{Uniform}(0, p(x^{(t)}))$

**Part 1:** Stepping Out

Sample interval  $(x_l, x_r)$  enclosing  $x^{(t)}$ .

$r \sim \text{Uniform}(u, w)$

$(x_l, x_r) = (x^{(t)} - r, x^{(t)} + w - r)$

Expand until endpoints are "outside" region under curve.

while( $\tilde{p}(x_l) > u$ )  $\{x_l = x_l - w\}$

while( $\tilde{p}(x_r) > u$ )  $\{x_r = x_r + w\}$

**Part 2:** Sample  $x$  (Shrinking)

Draw  $x$  from within the interval  $(x_l, x_r)$ , then accept or shrink.

# Slice Sampling

## Algorithm:

**Goal:** sample  $(x, u)$  given  $(u^{(t)}, x^{(t)})$ .

$u \sim \text{Uniform}(0, p(x^{(t)}))$

**Part 1: Stepping Out**

Sample interval  $(x_l, x_r)$  enclosing  $x^{(t)}$ .

$r \sim \text{Uniform}(u, w)$

$(x_l, x_r) = (x^{(t)} - r, x^{(t)} + w - r)$

Expand until endpoints are "outside" region under curve.

while( $\tilde{p}(x_l) > u$ )  $\{x_l = x_l - w\}$

while( $\tilde{p}(x_r) > u$ )  $\{x_r = x_r + w\}$

**Part 2: Sample  $x$  (Shrinking)**

while(true) {

Draw  $x$  from within the interval  $(x_l, x_r)$ , then accept or shrink.

$x \sim \text{Uniform}(x_l, x_r)$

if( $\tilde{p}(x) > u$ ) {break}

else if( $x > x^{(t)}$ )  $\{x_r = x\}$

else  $\{x_l = x\}$

}

$x^{(t+1)} = x, u^{(t+1)} = u$

# Slice Sampling

Extra Slides

## Multivariate Distributions

- Resample each variable  $x_i$  **one-at-a-time** (just like Gibbs Sampling)
- Does not require sampling from
$$p(x_i | \{x_j\}_{j \neq i})$$
- Only need to evaluate a quantity **proportional** to the conditional

$$p(x_i | \{x_j\}_{j \neq i}) \propto \tilde{p}(x_i | \{x_j\}_{j \neq i})$$

# Hamiltonian Monte Carlo

Extra Slides

- Suppose we have a distribution of the form:

$$p(\boldsymbol{x}) = \exp\{-E(\boldsymbol{x})\}/Z$$

where  $\boldsymbol{x} \in \mathcal{R}^N$

- We could use **random-walk M-H** to draw samples, but it seems a shame to **discard gradient information**  $\nabla_{\boldsymbol{x}} E(\boldsymbol{x})$
- If we can evaluate it, the gradient tells us where to look for **high-probability regions!**

# Background: Hamiltonian Dynamics

Extra Slides

## Applications:

- Following the motion of atoms in a fluid through time
- Integrating the motion of a solar system over time
- Considering the evolution of a galaxy (i.e. the motion of its stars)
- “molecular dynamics”
- “N-body simulations”

## Properties:

- Total energy of the system  $H(x,p)$  stays constant
- Dynamics are reversible

Important for  
detailed balance



# Background: Hamiltonian Dynamics

Extra Slides

Let  $\mathbf{x} \in \mathcal{R}^N$  be a position

$\mathbf{p} \in \mathcal{R}^N$  be a momentum

Potential energy:  $E(\mathbf{x})$

Kinetic energy:  $K(\mathbf{p}) = \mathbf{p}^T \mathbf{p} / 2$

Total energy:  $H(\mathbf{x}, \mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p})$

Hamiltonian function

Given a starting position  $\mathbf{x}^{(l)}$  and a starting momentum  $\mathbf{p}^{(l)}$  we can simulate the Hamiltonian dynamics of the system via:

1. Euler's method
2. Leapfrog method
3. etc.

# Background: Hamiltonian Dynamics

Extra Slides

## Parameters to tune:

1. Step size,  $\epsilon$
2. Number of iterations,  $L$

## Leapfrog Algorithm:

for  $\tau$  in  $1 \dots L$ :

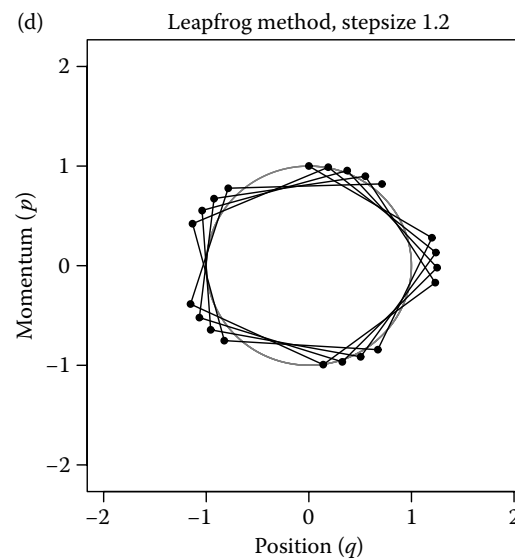
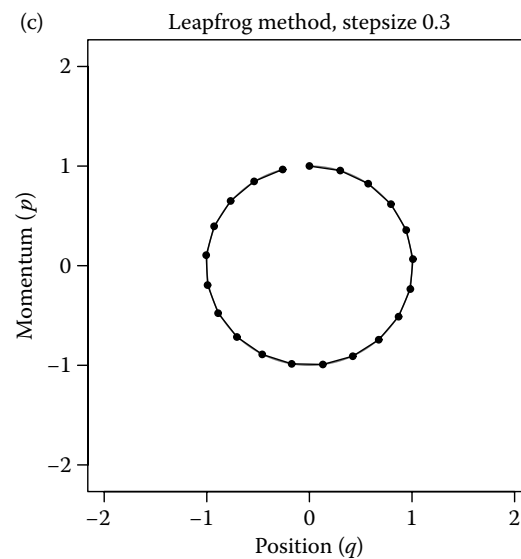
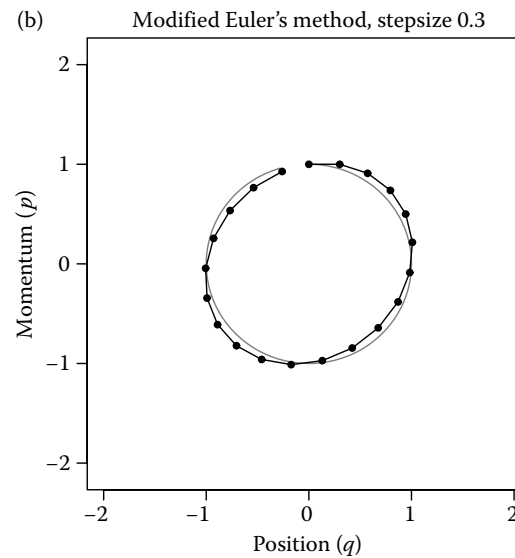
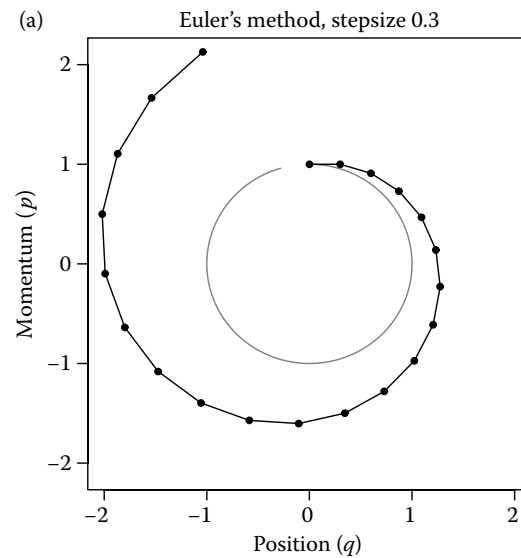
$$\mathbf{p} = \mathbf{p} - \frac{\epsilon}{2} \nabla_{\mathbf{x}} E(\mathbf{x})$$

$$\mathbf{x} = \mathbf{x} + \epsilon \mathbf{p}$$

$$\mathbf{p} = \mathbf{p} - \frac{\epsilon}{2} \nabla_{\mathbf{x}} E(\mathbf{x})$$

# Background: Hamiltonian Dynamics

Extra Slides



# Hamiltonian Monte Carlo

## Preliminaries

**Goal:**  $p(\mathbf{x}) = \exp\{-E(\mathbf{x})\}/Z$  where  $\mathbf{x} \in \mathcal{R}^N$

**Define:**

$$K(\mathbf{p}) = \mathbf{p}^T \mathbf{p} / 2$$

$$H(\mathbf{x}, \mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p})$$

$$\begin{aligned} p(\mathbf{x}, \mathbf{p}) &= \exp\{-H(\mathbf{x}, \mathbf{p})\} / Z_H \\ &= \exp\{-E(\mathbf{x})\} \exp\{-K(\mathbf{p})\} / Z_H \end{aligned}$$

**Note:**

Since  $p(\mathbf{x}, \mathbf{p})$  is separable...

$$\Rightarrow \sum_{\mathbf{p}} p(\mathbf{x}, \mathbf{p}) = \exp\{-E(\mathbf{x})\} / Z$$

Target dist.

$$\Rightarrow \sum_{\mathbf{x}} p(\mathbf{x}, \mathbf{p}) = \exp\{-K(\mathbf{p})\} / Z_K$$

Gaussian

# Whiteboard

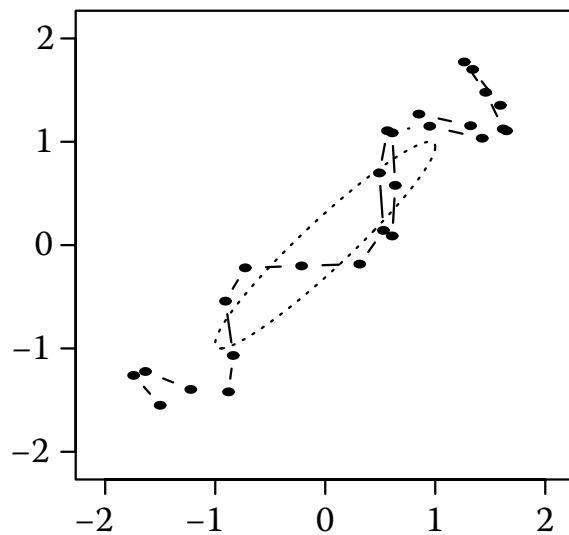
*Extra Slides*

- Hamiltonian Monte Carlo algorithm  
(aka. Hybrid Monte Carlo)

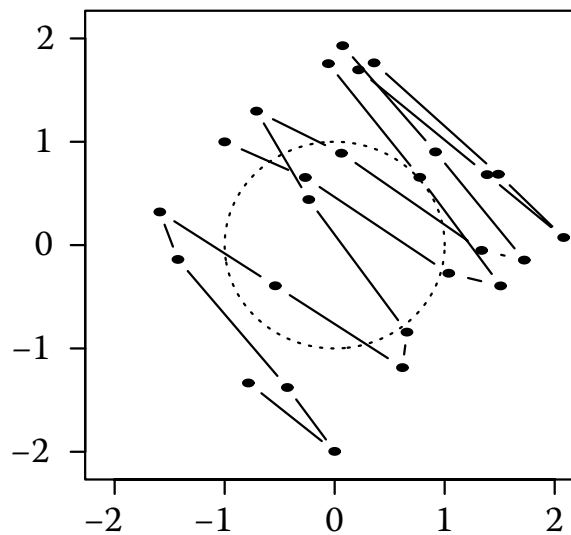
# Hamiltonian Monte Carlo

Extra Slides

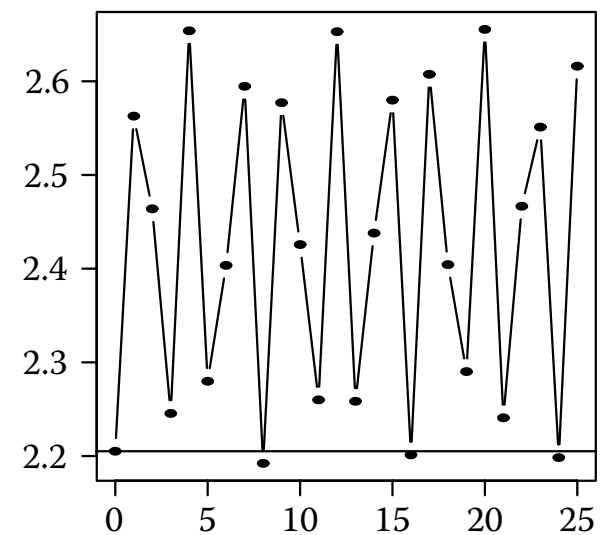
Position coordinates



Momentum coordinates

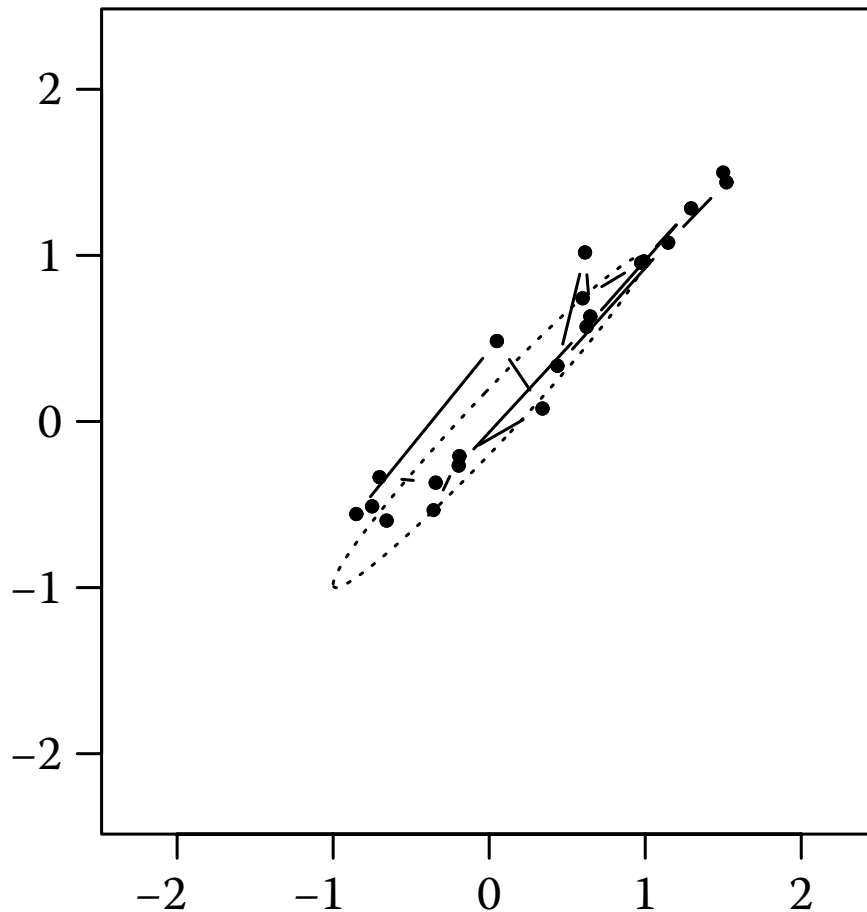


Value of Hamiltonian

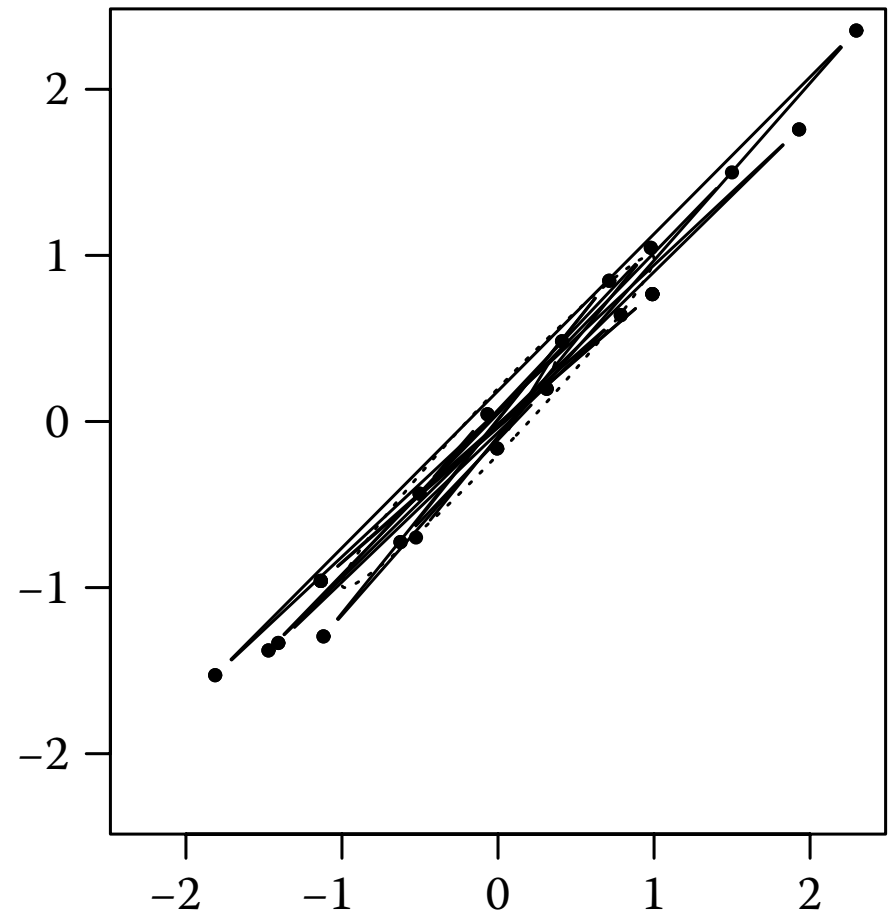


# M-H vs. HMC

Random-walk Metropolis



Hamiltonian Monte Carlo



# **SUPERVISED TRAINING WITH GIBBS SAMPLING**



# Motivation: Graphical Models

- Most recent advancements in NLP come from better **text input** representation from modern neural architectures
- Graphical models provide expressive modeling of the **output label space**

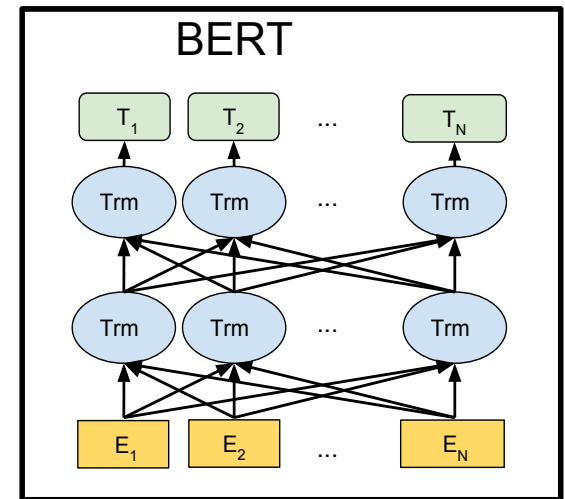
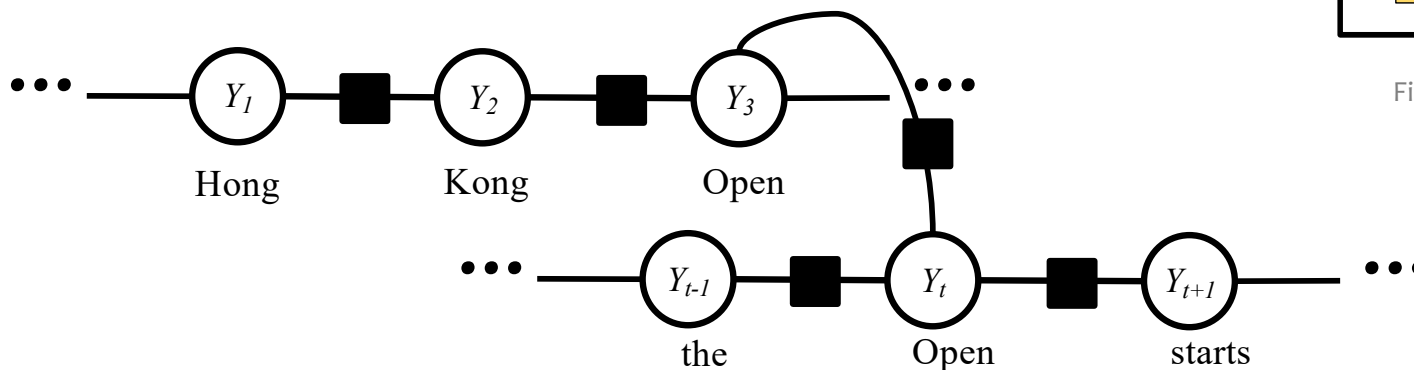
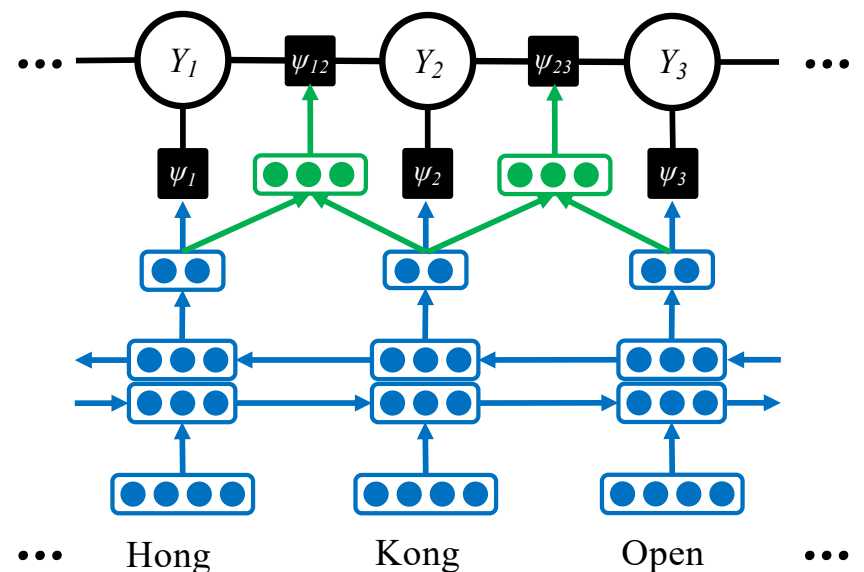


Figure from Devlin et al. (2018)

# Background: Linear-chain CRF

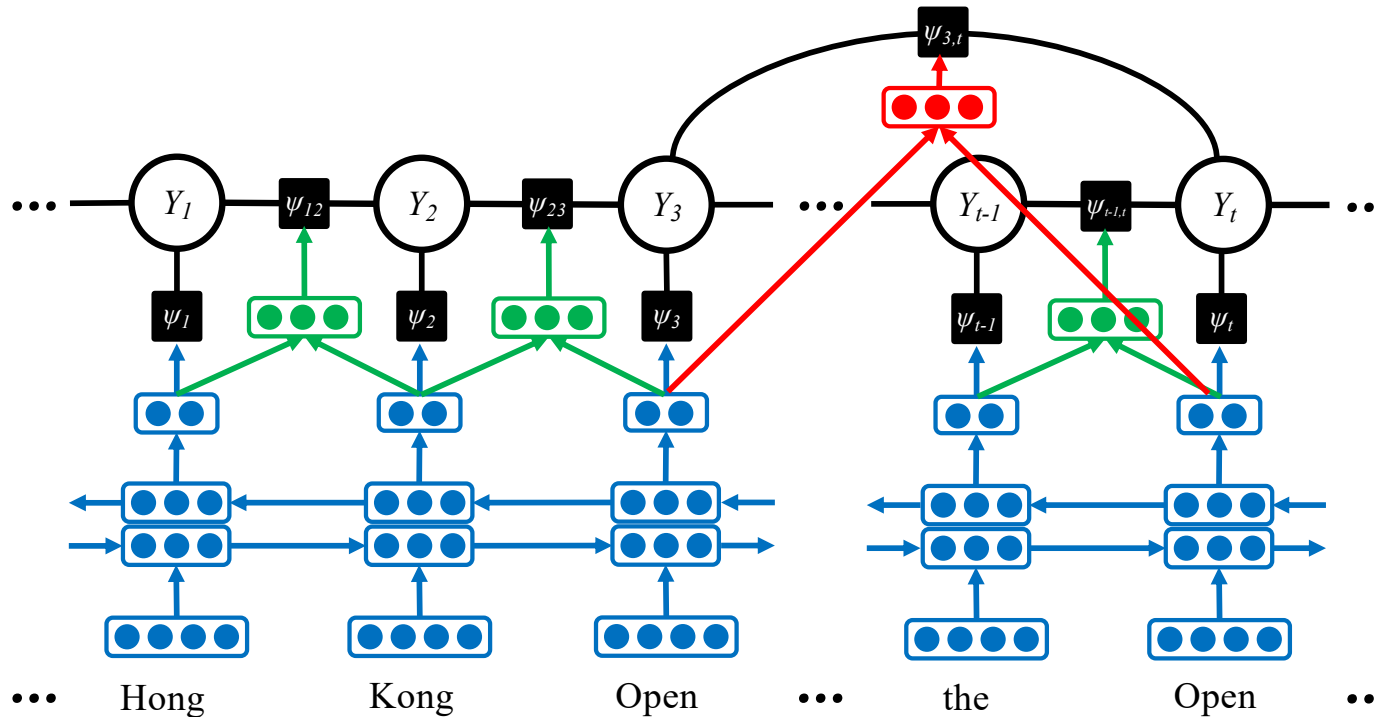
- Prior state-of-the-art approaches for sequence labeling have adopted linear-chain CRFs
  - Model bi-gram dependencies of adjacent labels
  - Exact inference can be done in polynomial time with forward-backward and Viterbi
- We are interested in more complex and expressive CRFs
  - Exact inference may no longer be affordable



Neural linear-chain CRF seen in e.g. Lample et al., 2016; Yang et al., 2016

# Skip-chain CRFs for NER

- Different occurrences of the same token often have the same label
- Skip-chains: long-range factors connecting recurring tokens



# Inference for Neural CRFs

- A neural CRF defines a conditional distribution:

$$p(y|x; \Theta) = \frac{\exp(s(x, y; \Theta))}{\sum_{y' \in \mathcal{Y}(x)} \exp(s(x, y'; \Theta))}$$

- Training time inference: compute the partition function
- Inference time: find the output with the highest probability

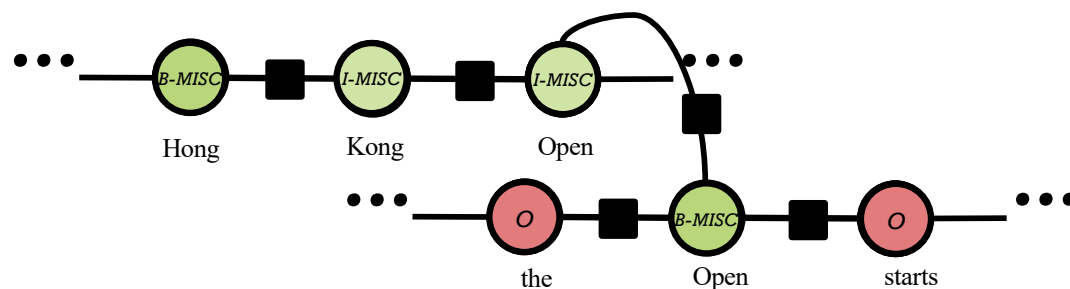
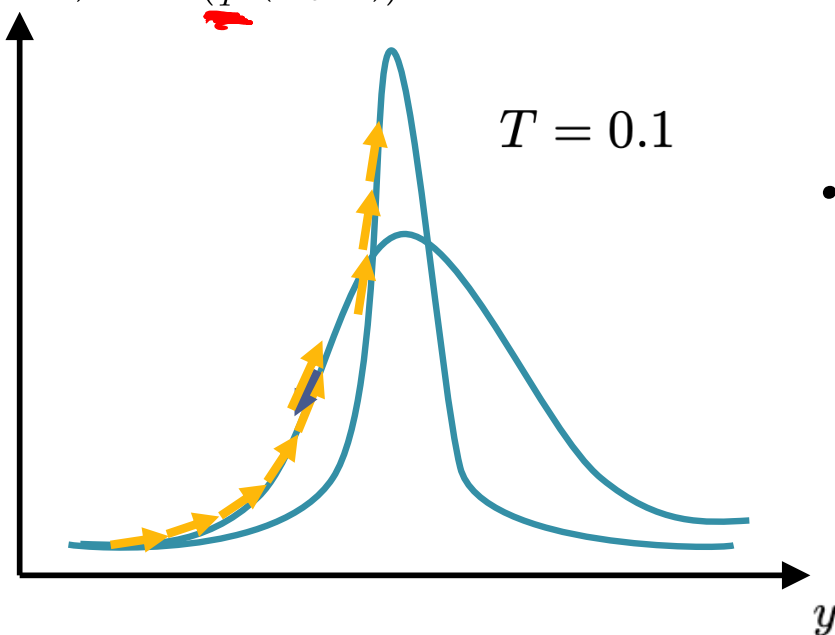
$$\hat{y} = \arg \max_{y \in \mathcal{Y}(x)} s(x, y; \Theta)$$

# Inference for Neural CRFs

✓ *deterministic annealing for BP.*

- Approximate inference: Gibbs sampling with annealing
- Gibbs sampling decoding is a local search algorithm for the maxima

$$p(y|x; \Theta, T) \propto \exp\left(\frac{1}{T}s(x, y; \Theta)\right)$$



# Computational Efficiency

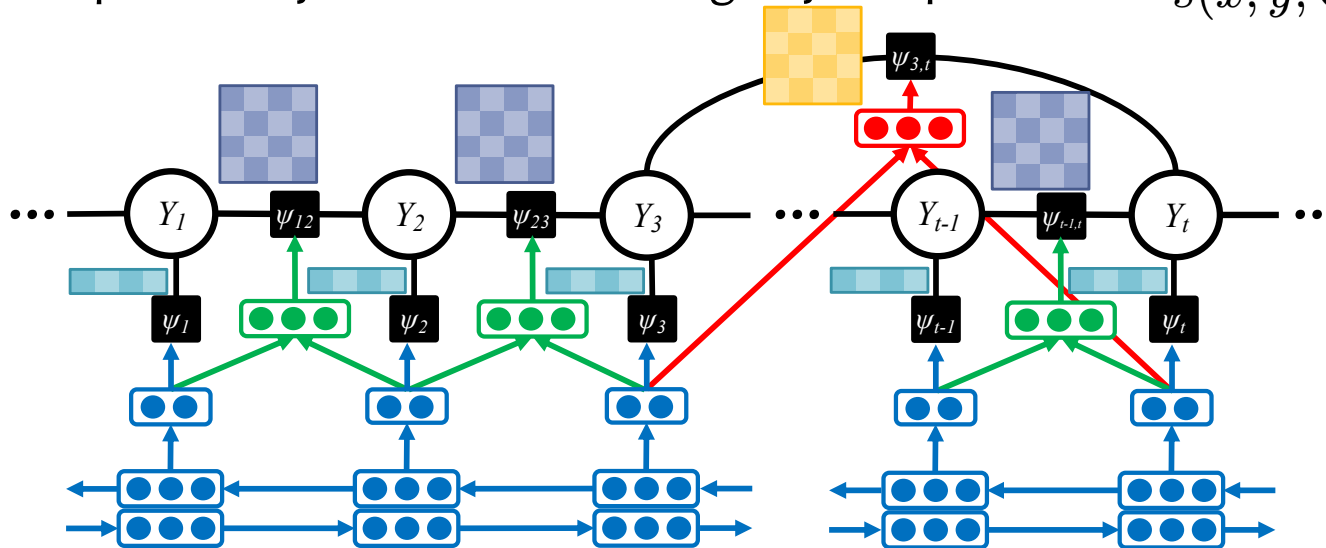
- Decompose the scoring function for computational efficiency

- Neural net component:

- Expensive to compute, but only depends on the input
- Computes only once before taking any samples

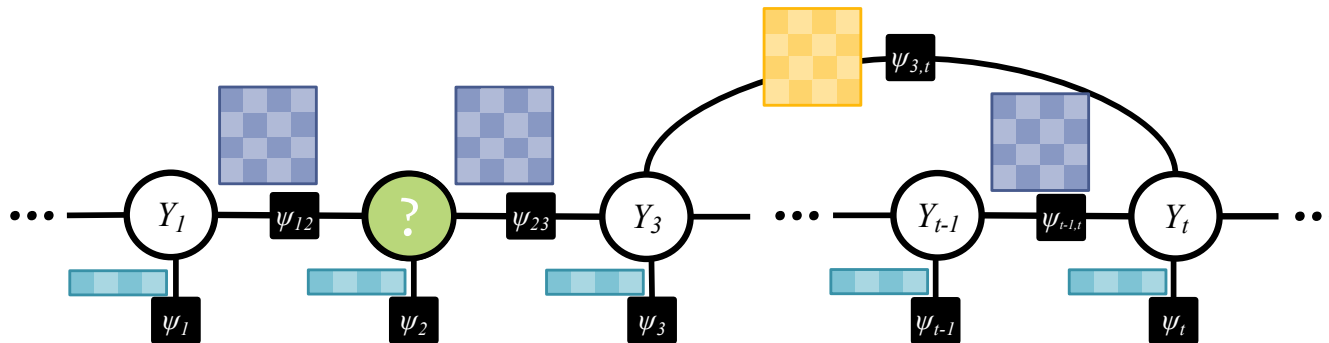
$$z = f(x; \theta_N)$$

$$s(x, y; \Theta) = s(x, y, z; \theta_G)$$



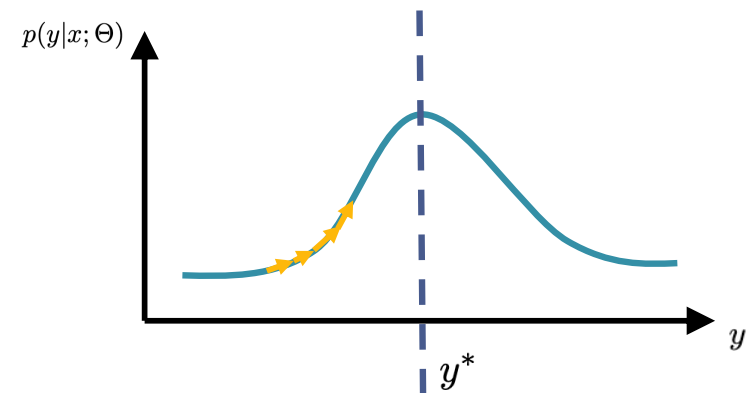
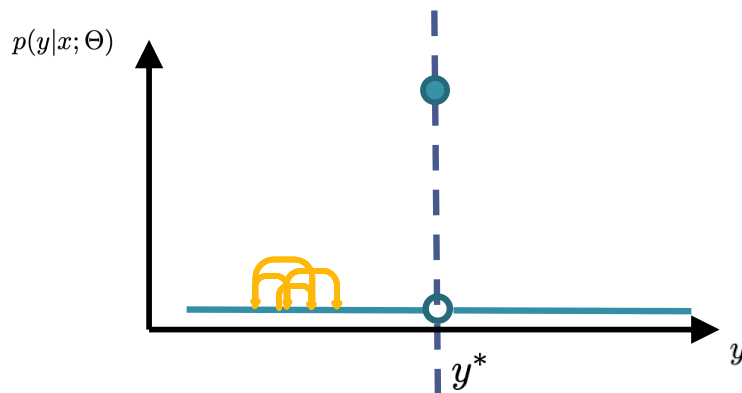
# Computational Efficiency

- Decompose the scoring function for computational efficiency
  - Graphical model component:
    - Depends on both input and output, but cheap to compute  $z = f(x; \theta_N)$
    - Local computation to take each sample
- $s(x, y; \Theta) = s(x, y, z; \theta_G)$



# Training for Gibbs Sampling

- Vanilla MLE only enforces a high score on the ground truth output
  - Extreme worst case: uniform low scores for all incorrect outputs
- An ideal scoring function should be able to differentiate between incorrect outputs, to guide the local search





# Neural SampleRank (NSR)

- Training objective: for each pair of outputs, the one with higher quality (i.e. closer to ground truth) also gets higher score

$$\ell(y_i, y_j) = [\Delta_\omega(y_i, y_j) - (s(y^+, x; \theta) - s(y^-, x; \theta))]_+$$

$\Delta_\omega(y_i, y_j) = \omega(y^+) - \omega(y^-)$

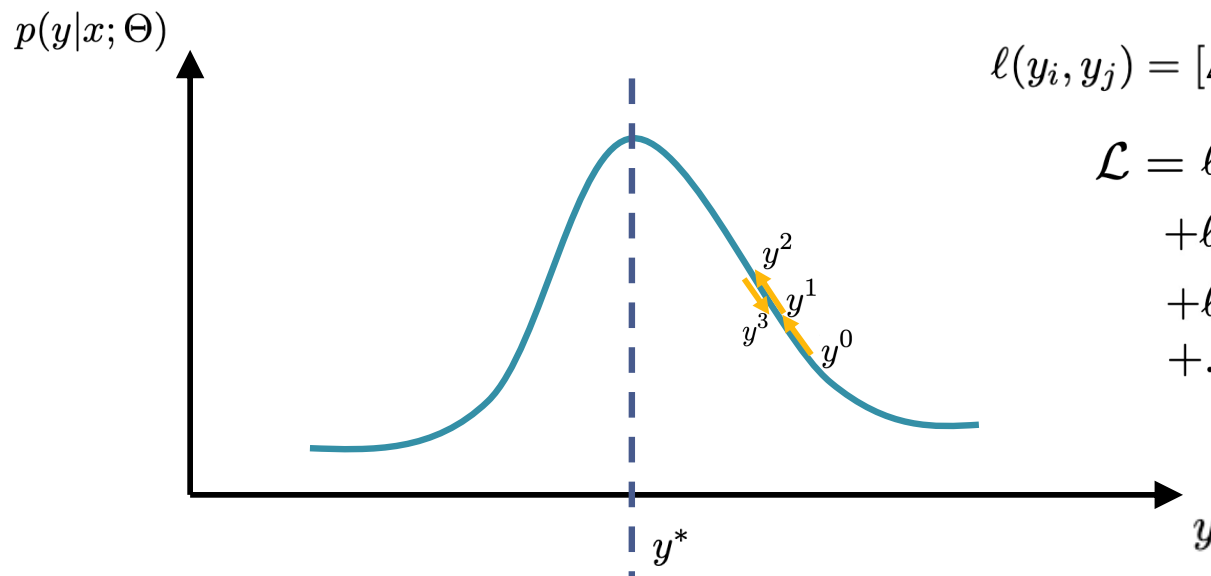
Metric (e.g. negative hamming)  
 $y^* = \arg \max_{y \in \mathcal{Y}} \omega(y)$

$y^+ = \arg \max_{y \in \{y_i, y_j\}} \omega(y)$

$y^- = \arg \min_{y \in \{y_i, y_j\}} \omega(y)$

# Neural SampleRank (NSR)

- The loss is accumulated across a sequence of samples during training
- A full inference is not needed
- Compared to SampleRank (Wick et al., 2011), the loss can be easily used to train neural net scoring factors



$$\ell(y_i, y_j) = [\Delta_\omega(y_i, y_j) - (s(y^+, x; \theta) - s(y^-, x; \theta))]_+$$

$$\begin{aligned} \mathcal{L} = & \ell(y^0, y^1) + \ell(y^1, y^*) \\ & + \ell(y^1, y^2) + \ell(y^2, y^*) \\ & + \ell(y^2, y^3) + \ell(y^3, y^*) \\ & + \dots \end{aligned}$$

# Results: NER (CoNLL-02/03)

- Models with contextualized embeddings

Model	Learning	English F1	German F1	Dutch F1
ELMo (Peters et al., 2018)	MLE	92.22	----	----
BERT (Devlin et al., 2018)	MLE	92.80	----	----
Flair (Akbik et al., 2019)	MLE	<b>93.18</b>	88.27	90.44
Our baseline Flair	MLE	92.58	<b>88.30</b>	90.63
+ skip-chain CRF	NSR	92.56	87.97	<b>91.44*</b>

	English	German	Dutch
# token	204,567	207,484	202,931
# document	946	553	287
# skip-chain	29,309	31,683	44,309

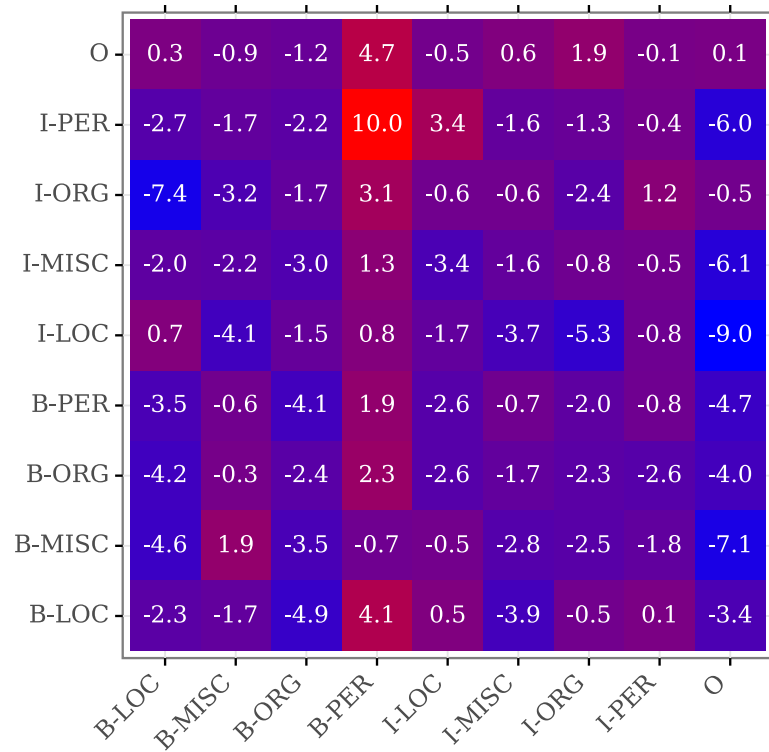
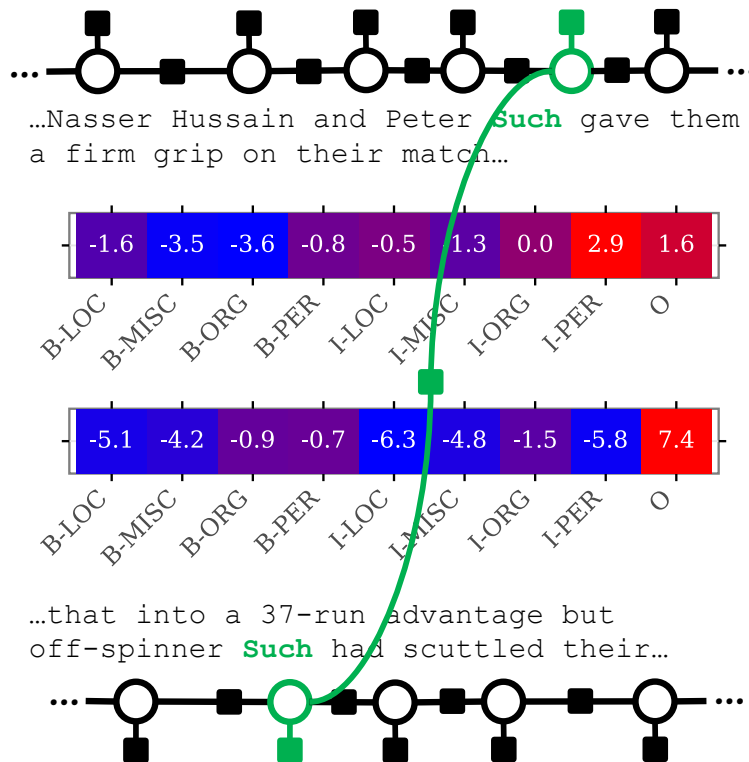
# Results: NER (CoNLL-02/03)

- Models without contextualized embeddings

Model	Learning	English F1
BiLSTM-CRF (Lample et al., 2016)	MLE	90.94
BiGRU-CRF (Yang et al., 2016)	MLE	91.20
Our baseline BiLSTM-CRF	MLE	91.01
+ skip-chain CRF	NSR	<b>91.68*</b>

Model	Learning	German F1
BiLSTM-CRF (Lample et al., 2016)	MLE	78.76
BiLSTM (Riedl and Padó, 2018)	MLE	82.99
Our baseline BiLSTM-CRF	MLE	83.55
+ skip-chain CRF	NSR	<b>84.50*</b>

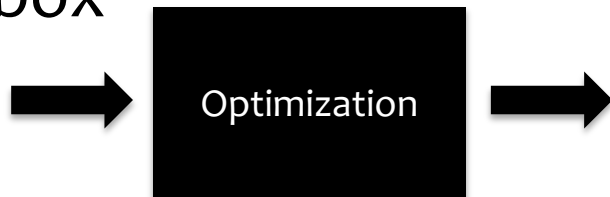
# Results: Qualitative Analysis



# **SUPERVISED LEARNING FOR BAYES NETS**

# Recipe for Gradient-based Learning

1. Write down the objective function
2. Compute the partial derivatives of the objective (i.e. gradient, and maybe Hessian)
3. Feed objective function and derivatives into black box



4. Retrieve optimal parameters from black box



- This is how we trained MRFs and CRFs
- The same approach also applies to Bayesian Networks
- We just compute the gradient of the Bayes Net's log-likelihood of the data

But sometimes <sup>for DGM</sup> there's an even easier way...

# **SUPERVISED LEARNING FOR BAYES NETS (BY “COUNTING”)**



# Recipe for Closed-form MLE

1. Assume data was generated i.i.d. from some model  
(i.e. write the generative story)  
$$x^{(i)} \sim p(x|\boldsymbol{\theta})$$
2. Write log-likelihood  
$$\ell(\boldsymbol{\theta}) = \log p(x^{(1)}|\boldsymbol{\theta}) + \dots + \log p(x^{(N)}|\boldsymbol{\theta})$$
3. Compute partial derivatives (i.e. gradient)  
$$\begin{aligned}\partial \ell(\boldsymbol{\theta}) / \partial \theta_1 &= \dots \\ \partial \ell(\boldsymbol{\theta}) / \partial \theta_2 &= \dots \\ &\dots \\ \partial \ell(\boldsymbol{\theta}) / \partial \theta_M &= \dots\end{aligned}$$
4. Set derivatives to zero and solve for  $\boldsymbol{\theta}$   
$$\partial \ell(\boldsymbol{\theta}) / \partial \theta_m = 0 \text{ for all } m \in \{1, \dots, M\}$$
  
$$\boldsymbol{\theta}^{\text{MLE}} = \text{solution to system of } M \text{ equations and } M \text{ variables}$$
5. Compute the second derivative and check that  $\ell(\boldsymbol{\theta})$  is concave down at  $\boldsymbol{\theta}^{\text{MLE}}$

# Machine Learning

The **data** inspires  
the structures  
we want to  
predict



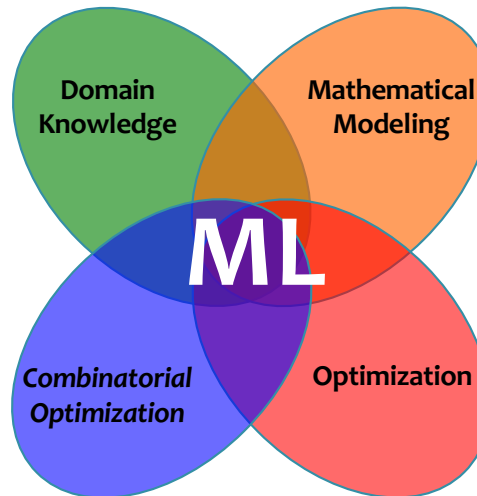
Our **model**  
defines a score  
for each structure

It also tells us  
what to optimize



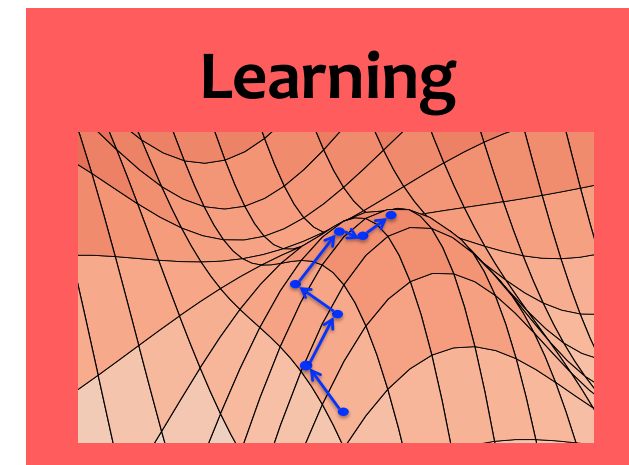
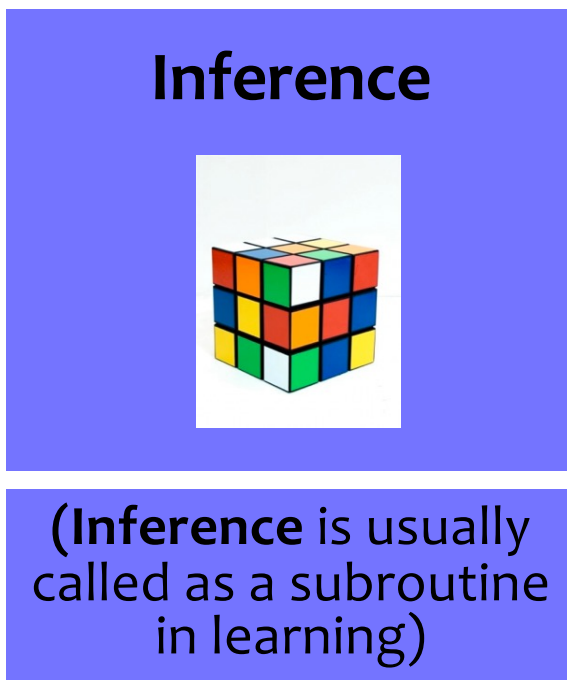
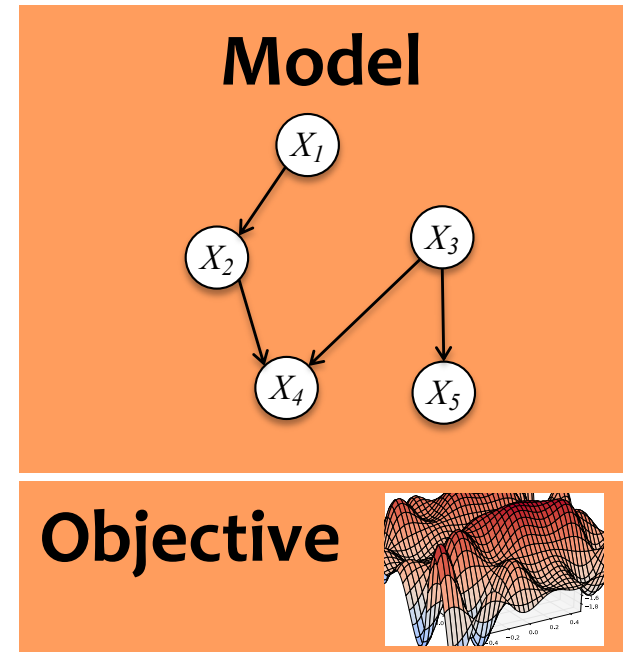
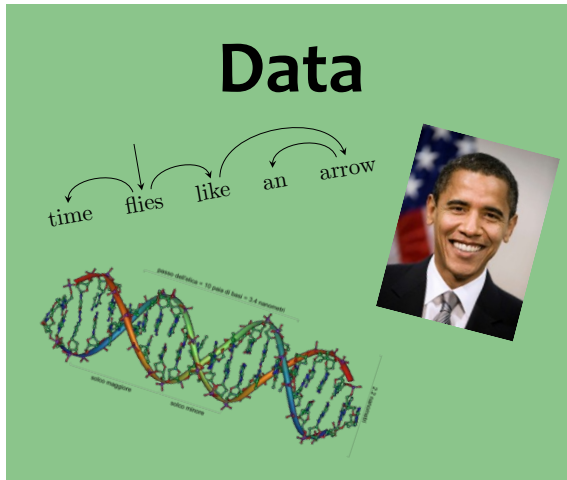
**Inference** finds  
{best structure, marginals,  
partition function} for a  
new observation

(**Inference** is usually  
called as a subroutine  
in learning)

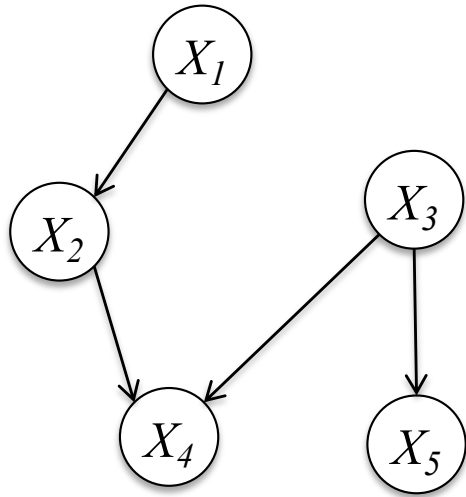


**Learning** tunes the  
parameters of the  
model

# Machine Learning

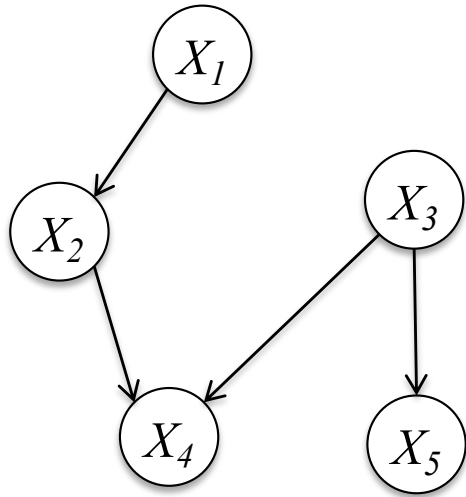


# Learning Fully Observed BNs



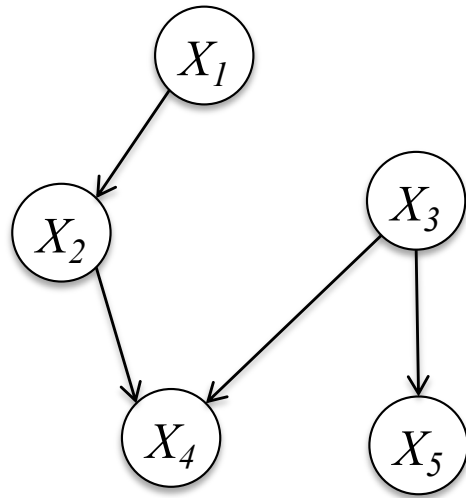
$$\begin{aligned} p(X_1, X_2, X_3, X_4, X_5) = \\ p(X_5|X_3)p(X_4|X_2, X_3) \\ p(X_3)p(X_2|X_1)p(X_1) \end{aligned}$$

# Learning Fully Observed BNs



$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

# Learning Fully Observed BNs

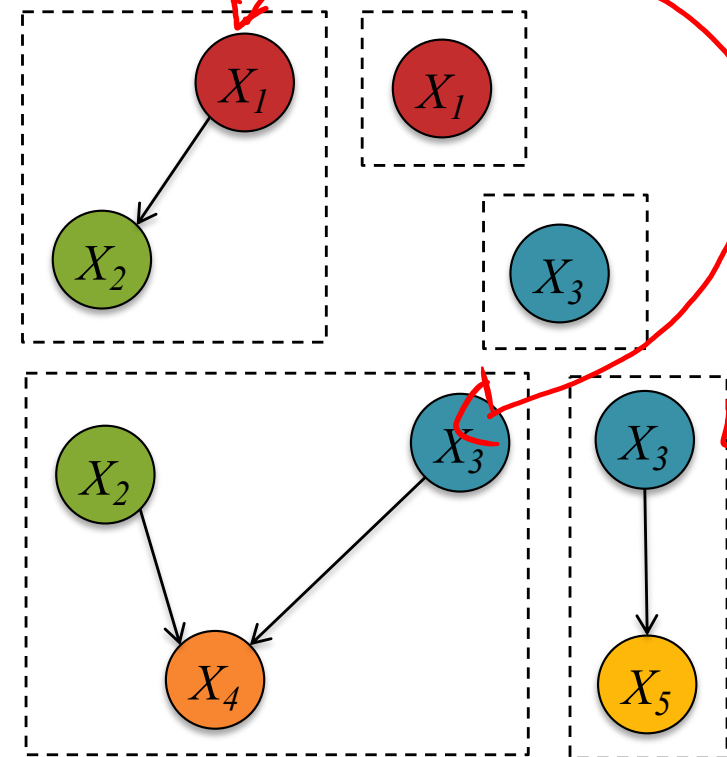
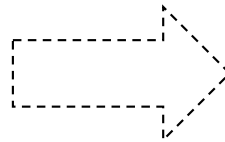
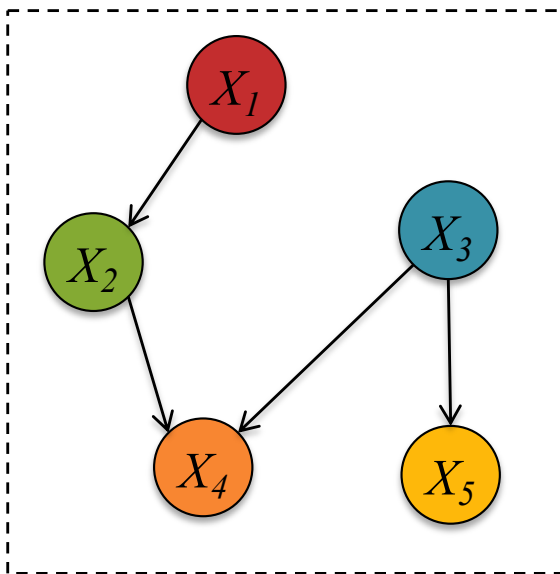


$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

How do we learn these **conditional** and **marginal** distributions for a Bayes Net?

# Learning Fully Observed BNs

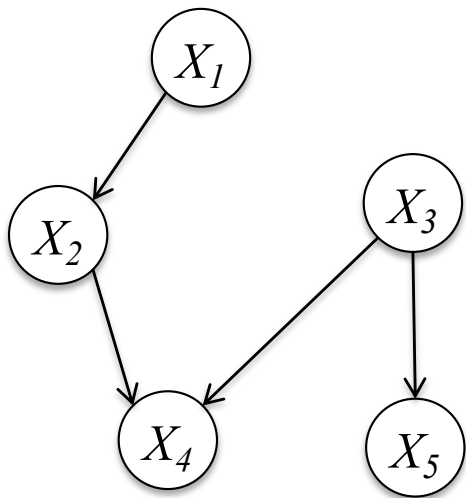
Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data



$$p(X_1, X_2, X_3, X_4, X_5) = p(X_5|X_3)p(X_4|X_2, X_3)p(X_3)p(X_2|X_1)p(X_1)$$

# Learning Fully Observed BNs

How do we learn these  
**conditional** and **marginal**  
distributions for a Bayes Net?



$$\begin{aligned}\theta^* &= \operatorname{argmax}_{\theta} \log p(X_1, X_2, X_3, X_4, X_5) \\ &= \operatorname{argmax}_{\theta} \log p(X_5|X_3, \theta_5) + \log p(X_4|X_2, X_3, \theta_4) \\ &\quad + \log p(X_3|\theta_3) + \log p(X_2|X_1, \theta_2) \\ &\quad + \log p(X_1|\theta_1)\end{aligned}$$

$$\theta_1^* = \operatorname{argmax}_{\theta_1} \log p(X_1|\theta_1)$$

$$\theta_2^* = \operatorname{argmax}_{\theta_2} \log p(X_2|X_1, \theta_2)$$

$$\theta_3^* = \operatorname{argmax}_{\theta_3} \log p(X_3|\theta_3)$$

$$\theta_4^* = \operatorname{argmax}_{\theta_4} \log p(X_4|X_2, X_3, \theta_4)$$

$$\theta_5^* = \operatorname{argmax}_{\theta_5} \log p(X_5|X_3, \theta_5)$$



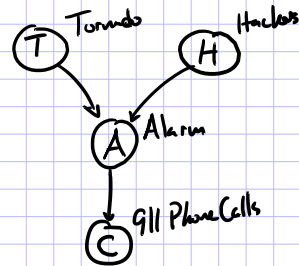
# Example: Tornado Alarms



1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

# Learning Fully Observed BNs

Ex: Tornado Alarms



$$\begin{aligned}
 H &\sim \text{Bernoulli}(\eta) \\
 T &\sim \text{Bernoulli}(\tau) \\
 A &\sim \text{Bernoulli}(\alpha_{H,T}) \\
 C &\sim \text{Uniform}(\{1, \dots, 63\}) + A * \text{Uniform}(\{1, \dots, 63\})
 \end{aligned}$$

parameters (pointing to  $\eta, \tau, \alpha_{H,T}$ )  
 no parameters (pointing to the Uniform distributions)  
 integer (pointing to the set notation in the last line)

Dataset

i	T	H	A	C
1	0	0	0	2
2	0	0	0	6
3	0	0	0	4
...	1	0	0	3
...	1	0	0	1
...	1	0	1	10
...	1	0	1	7
...	0	1	0	2
...	0	1	1	12
...	0	1	0	5
...	1	1	1	10
12	1	0	0	2

MLEs in Closed Form

$$\begin{aligned}
 l(\eta, \tau, \alpha) &= \log \prod_{i=1}^N p(t^{(i)}, h^{(i)}, a^{(i)}, c^{(i)} | \eta, \tau, \alpha) \\
 &= \sum_{i=1}^N \log p(t^{(i)} | \tau) + \log p(h^{(i)} | \eta) \\
 &\quad + \log p(a^{(i)} | t^{(i)}, h^{(i)}, \alpha) + \log p(c^{(i)} | a^{(i)})
 \end{aligned}$$

$$\hat{\eta}, \hat{\tau}, \hat{\alpha} = \arg \max_{\eta, \tau, \alpha} l(\eta, \tau, \alpha)$$

$$\hat{\eta} = \arg \max_{\eta} \sum_{i=1}^N \log p(h^{(i)} | \eta) = \#(H=1) / N$$

$$\hat{\tau} = \arg \max_{\tau} \sum_{i=1}^N \log p(t^{(i)} | \tau) = \#(T=1) / N$$

$$\hat{\alpha} = \arg \max_{\alpha} \sum_{i=1}^N \log p(a^{(i)} | t^{(i)}, h^{(i)}, \alpha)$$

What are the MLEs?

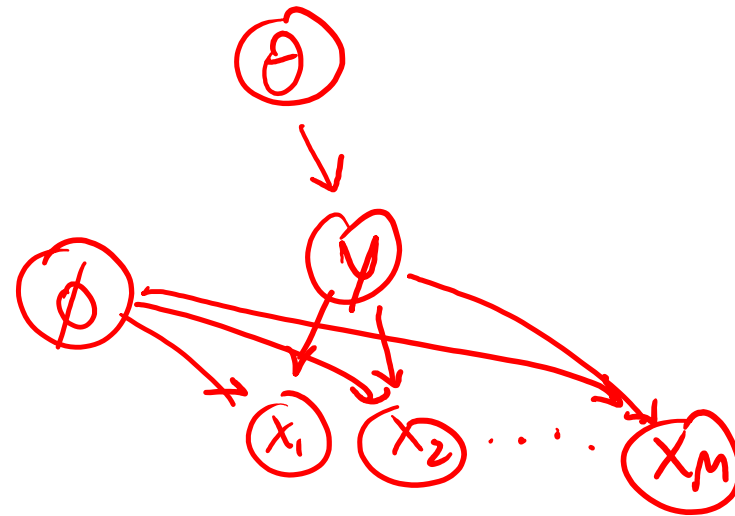
$$\hat{\eta} = 1/3$$

$$\hat{\tau} = 1/2$$

$$\hat{\alpha} =$$

	H=0	H=1
T=0	0	1/3
T=1	2/3	1

$$\hat{\alpha}_{t,h} = \frac{\#(A=1, T=t, H=h)}{\#(T=t, H=h)}$$



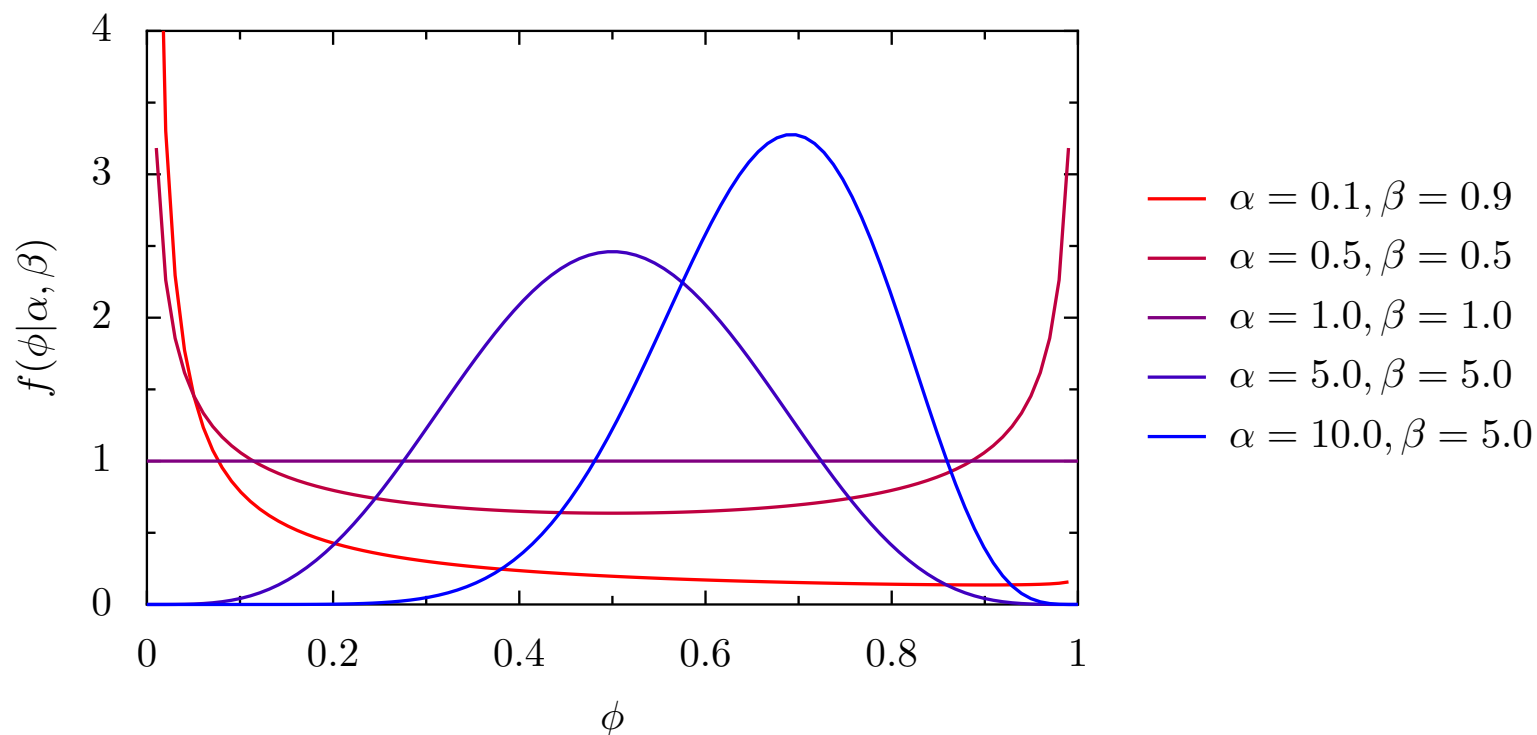
# BAYESIAN INFERENCE FOR NAÏVE BAYES

# Beta-Bernoulli Model

- Beta Distribution

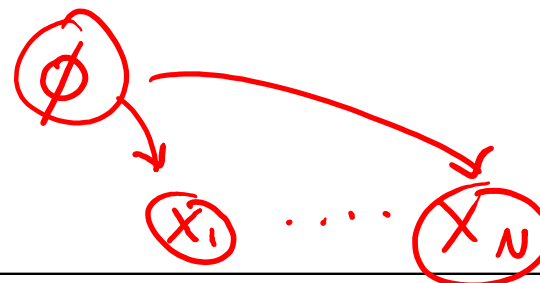
$$\phi \in [0, 1]$$

$$f(\phi|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$



# Beta-Bernoulli Model

- Generative Process



$$\phi \sim \text{Beta}(\alpha, \beta)$$

*[draw distribution over words]*

For each word  $n \in \{1, \dots, N\}$

$$x_n \sim \text{Bernoulli}(\phi)$$

*[draw word]*

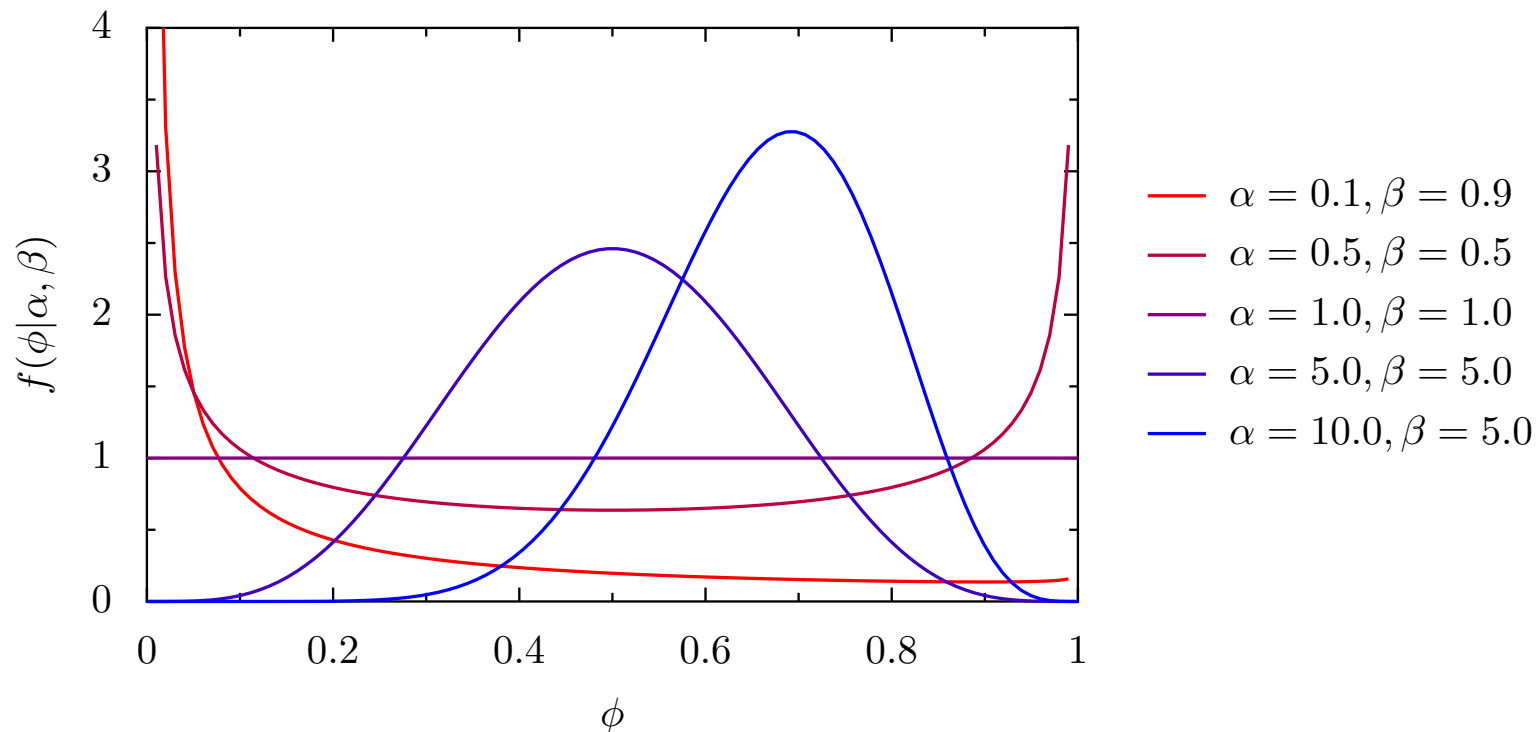
- Example corpus (heads/tails)

H	T	T	H	H	T	T	H	H	H
$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$

# Dirichlet-Multinomial Model

- Dirichlet Distribution

$$f(\phi|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

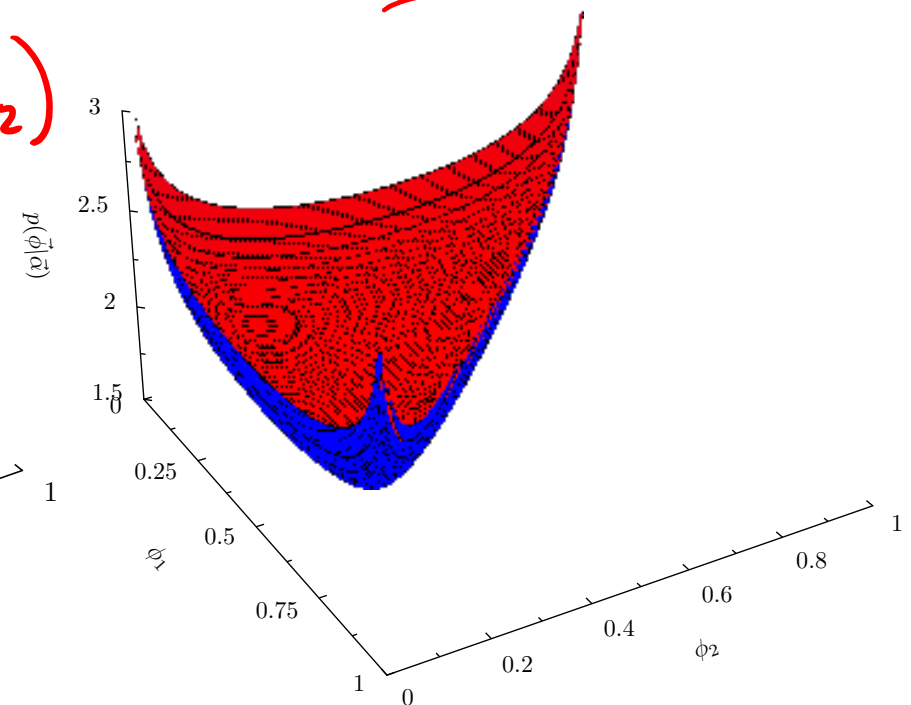
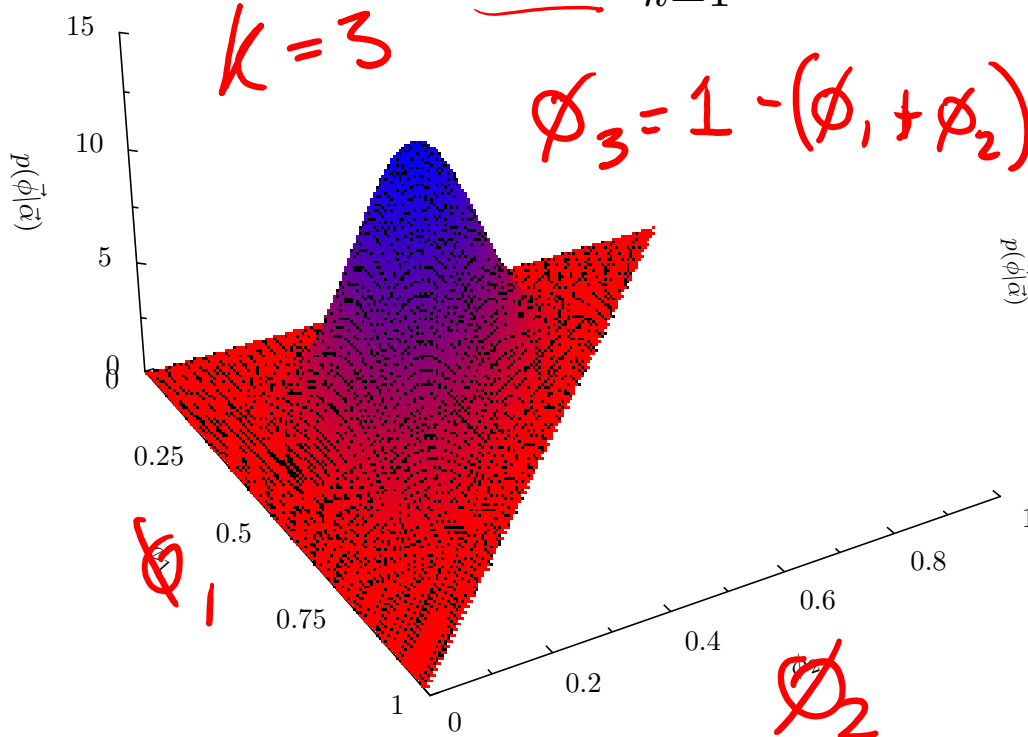


# Dirichlet-Multinomial Model

- Dirichlet Distribution

$$\vec{\phi} \in [0, 1]^K \quad \text{s.t.} \quad 1 = \sum_{k=1}^K \phi_k$$

$$p(\vec{\phi} | \underline{\alpha}) = \frac{1}{\underline{B(\alpha)}} \prod_{k=1}^K \phi_k^{\underline{\alpha_k} - 1} \quad \text{where} \quad \underline{B(\alpha)} = \frac{\prod_{k=1}^K \Gamma(\underline{\alpha_k})}{\Gamma(\sum_{k=1}^K \underline{\alpha_k})}$$



# Dirichlet-Multinomial Model

- Generative Process

$$\phi \sim \text{Dir}(\beta)$$

*[draw distribution over words]*

For each word  $n \in \{1, \dots, N\}$

$$x_n \sim \text{Mult}(1, \phi)$$

*[draw word]*

- Example corpus

the	he	is	the	and	the	she	she	is	is
$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$



# Dirichlet-Multinomial Model

The Dirichlet is **conjugate** to the Multinomial

$$\phi \sim \text{Dir}(\beta)$$

[draw distribution over words]

For each word  $n \in \{1, \dots, N\}$

$$x_n \sim \text{Mult}(1, \phi)$$

[draw word]

- The posterior of  $\phi$  is  $p(\phi|X) = \frac{p(X|\phi)p(\phi)}{P(X)}$
- Define the count vector  $\mathbf{n}$  such that  $n_t$  denotes the number of times word  $t$  appeared
- Then the posterior is also a Dirichlet distribution:  
 $p(\phi|X) \sim \text{Dir}(\beta + \mathbf{n})$

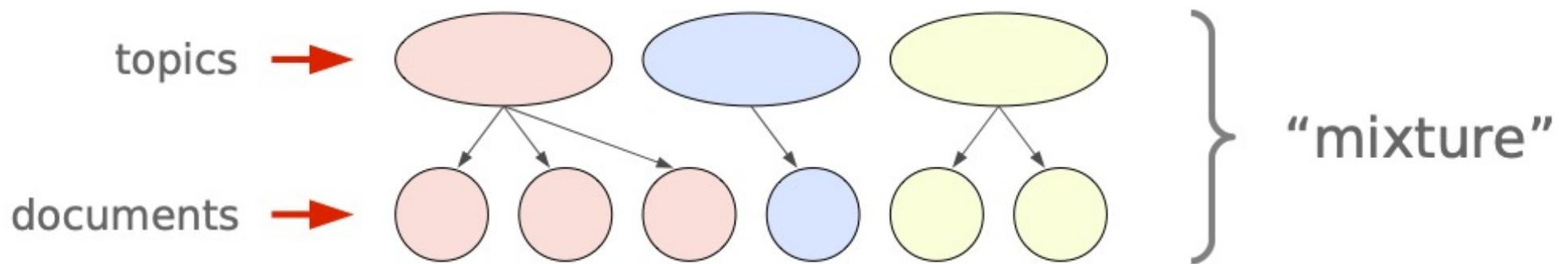
$$\begin{aligned}
 p(\vec{\phi} | \vec{x}, \vec{\beta}) &\propto p(\vec{x} | \vec{\phi}) p(\vec{\phi}) \\
 &= \left[ \prod_{i=1}^N p(x^{(i)} | \vec{\phi}) \right] p(\vec{\phi}) \\
 &\propto \left[ \prod_{k=1}^K \phi_k^{\beta_k} \right] \left[ \prod_{i=1}^N \prod_{k=1}^K \phi_k^{\mathbb{1}(x^{(i)}=k)} \right] \\
 &= \prod_{k=1}^K \phi_k^{\left[ \beta_k - 1 + \sum_{i=1}^N \mathbb{1}(x^{(i)}=k) \right]}
 \end{aligned}$$

$$\Rightarrow p(\vec{\phi} | \vec{x}, \vec{\beta}) \sim \text{Dirichlet}(\vec{\beta} + \vec{n})$$

where  $n_k = \# \text{ times } x^{(i)} = k$

# Dirichlet-Multinomial Mixture Model

- Generative Process



- Example corpus

the	he	is
$x_{11}$	$x_{12}$	$x_{13}$

Document 1

the	and	the
$x_{21}$	$x_{22}$	$x_{23}$

Document 2

she	she	is	is
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$

Document 3

# Dirichlet-Multinomial Mixture Model

- Generative Process

For each topic  $k \in \{1, \dots, K\}$ :

$$\phi_k \sim \text{Dir}(\beta)$$

*[draw distribution over words]*

$$\theta \sim \text{Dir}(\alpha)$$

*[draw distribution over topics]*

For each document  $m \in \{1, \dots, M\}$

$$z_m \sim \text{Mult}(1, \theta)$$

*[draw topic assignment]*

For each word  $n \in \{1, \dots, N_m\}$

$$x_{mn} \sim \text{Mult}(1, \phi_{z_m})$$

*[draw word]*

- Example corpus

the	he	is
$x_{11}$	$x_{12}$	$x_{13}$

Document 1

the	and	the
$x_{21}$	$x_{22}$	$x_{23}$

Document 2

she	she	is	is
$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$

Document 3

# Bayesian Inference for Naïve Bayes

## ***Whiteboard:***

- Naïve Bayes is not Bayesian
- What if we observed both words and topics?
- Dirichlet-Multinomial in the fully observed setting is just Naïve Bayes
- Three ways of estimating parameters:
  1. MLE for Naïve Bayes
  2. MAP estimation for Naïve Bayes
  3. Bayesian parameter estimation for Naïve Bayes