10-418/10-618 Machine Learning for Structured Data

Machine Learning Department
School of Computer Science
Carnegie Mellon University

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

Figure from Bishop (2006)

Random Walk Behavior of M-H

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

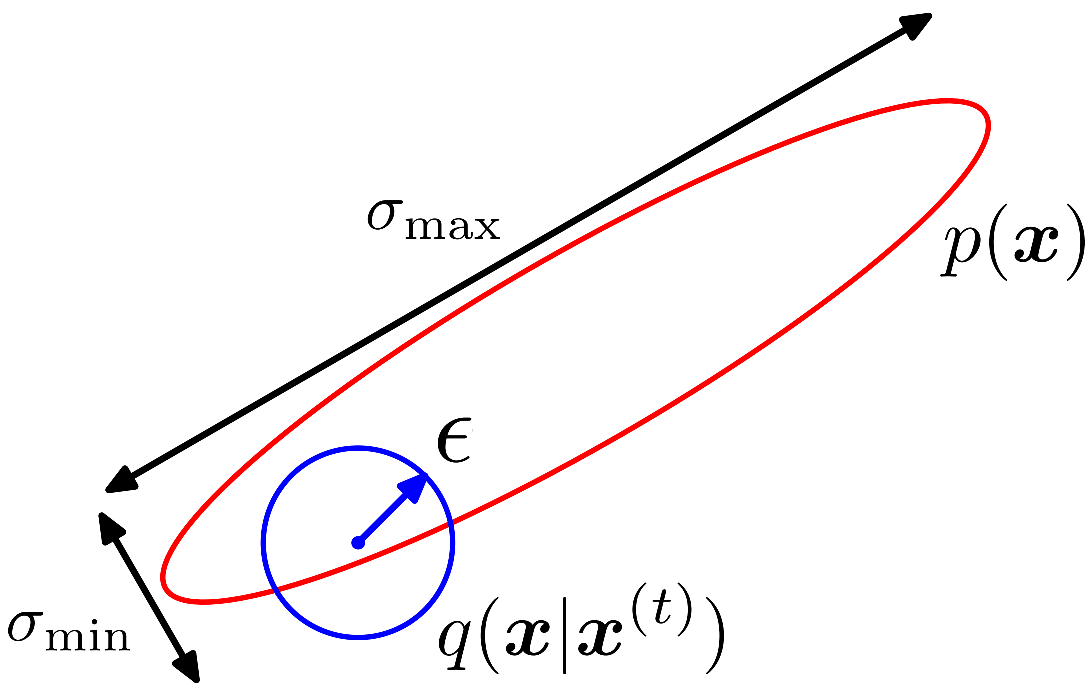
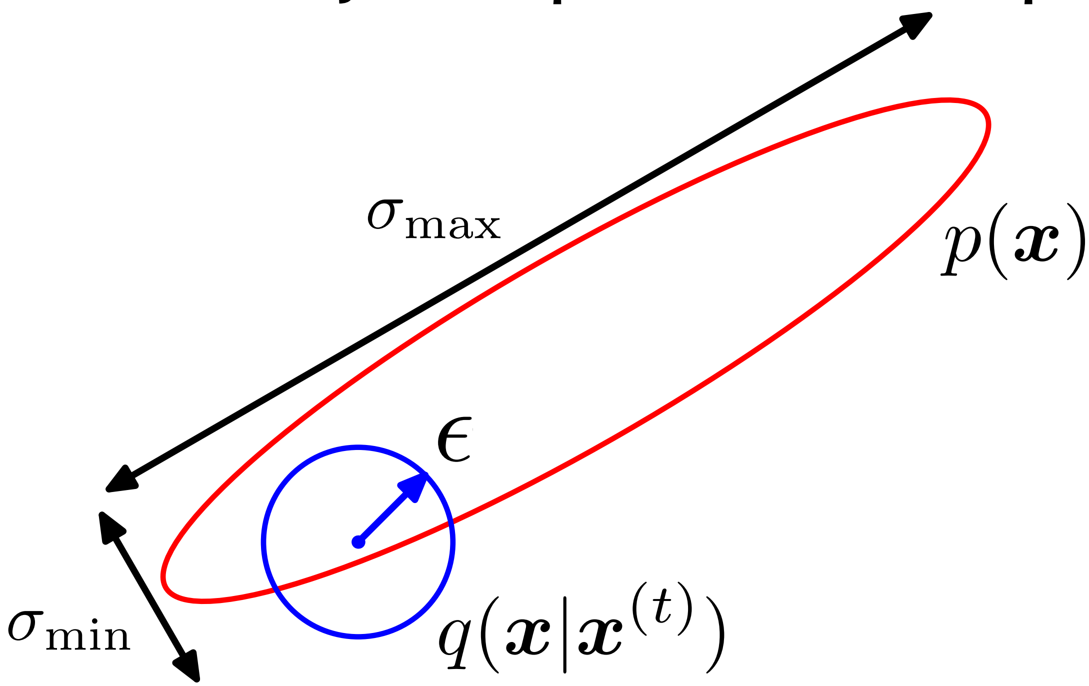


Figure from Bishop (2006)

Random Walk Behavior of M-H

- For Rejection Sampling, the accepted samples are 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 $q(\vec{x} \mid \vec{x}^{(t)}) \triangleq \begin{cases} x_i \sim p(x_i \mid \vec{x}_{\neg i}^{(t)}) \\ x_{\neg i} = x_{\neg i}^{(t)} \end{cases}$ where $i \sim \text{Unif}(1, \dots, J)$

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

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

$p(a,b) = p(a \mid b)\, p(b)$

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

1 1 1

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

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

MCMC PRACTICAL ISSUES

Practical Issues

- **Question:** Is it better to move along one dimension or many?

- **Answer:** For Metropolis-Hasings, 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 $y_1, y_2, \ldots, y_J \sim p(y_1, y_2, \ldots, y_J)$

Algorithm:

- Initialize $y_1, y_2, \ldots, y_J$ to arbitrary values
- For $t = 1, 2, \ldots$:

  for b in B: where $b \subseteq \{1, \ldots, J\}$

  $$y_b \sim p(y_b \mid y_{\neg b})$$

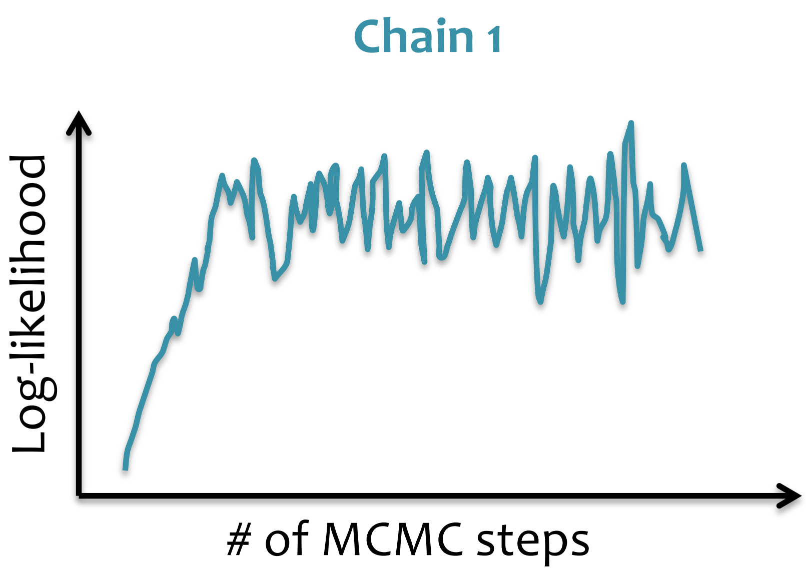
- Example: B = set of factors in a factor graph

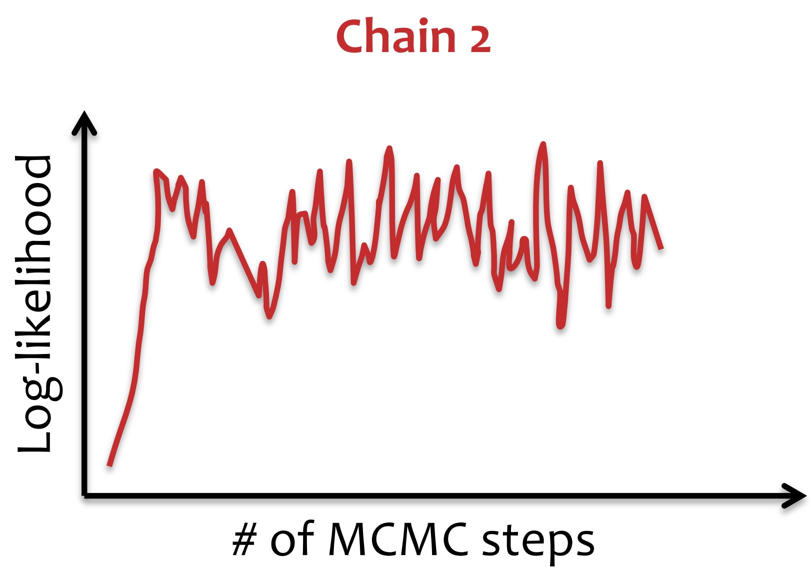
Why use blocks?

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

Practical Issues

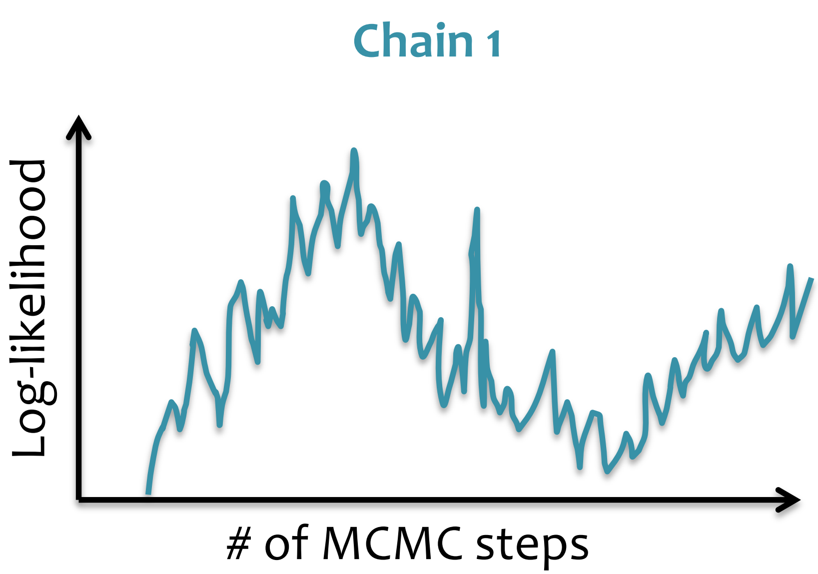
- **Question:** How do we assess convergence of the Markov chain?
- **Answer:** It's not easy!
  - Compare statistics of multiple independent chains
  - Ex: Compare log-likelihoods

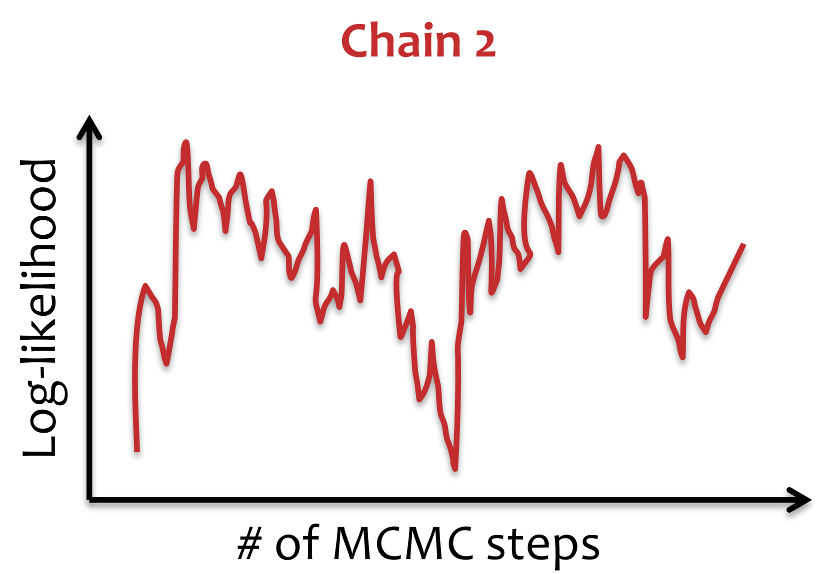




Practical Issues

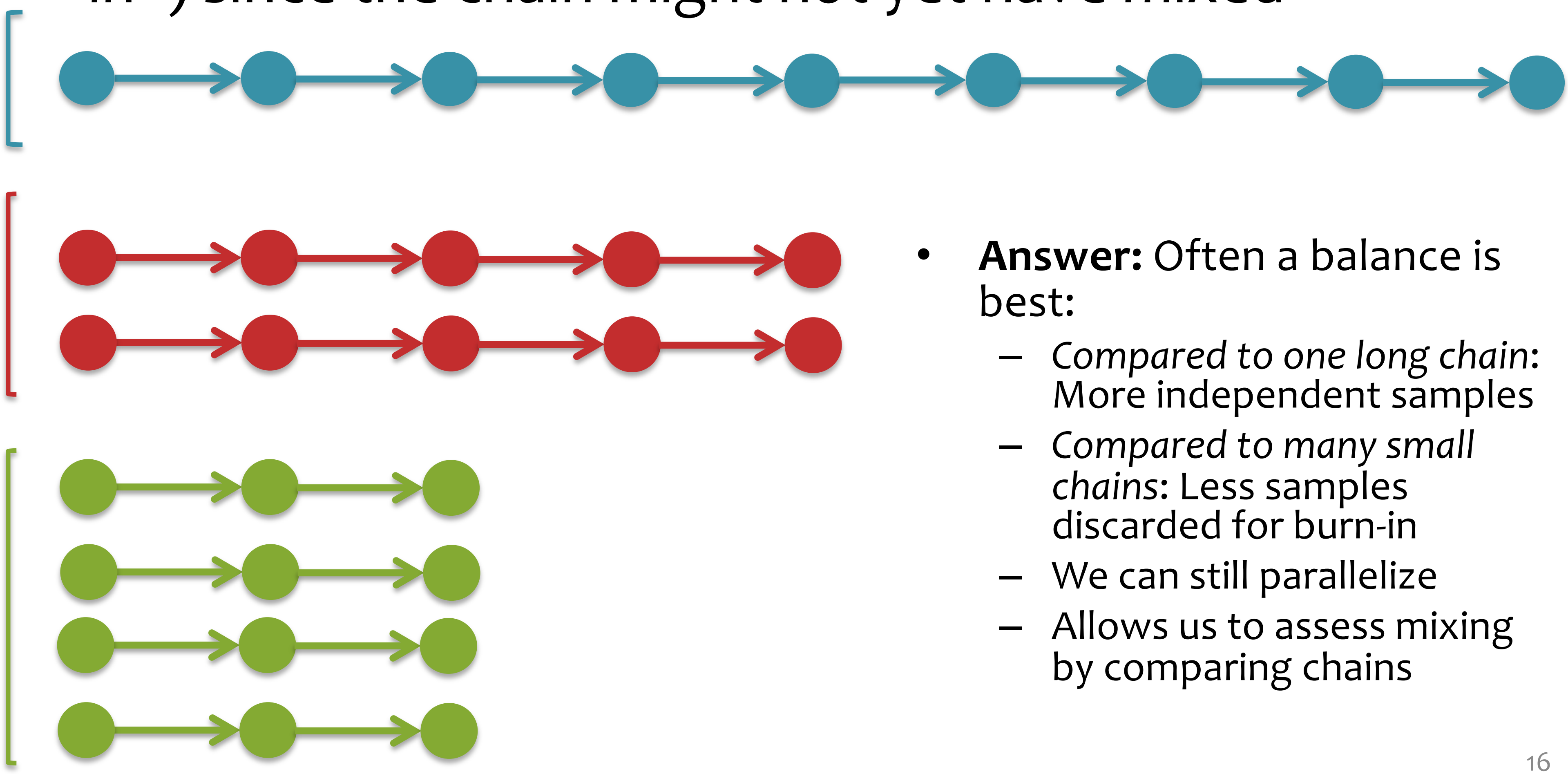
- **Question:** How do we assess convergence of the Markov chain?
- **Answer:** It's not easy!
  - Compare statistics of multiple independent chains
  - Ex: Compare log-likelihoods





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 $t \to \infty$
- **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

we're focused on first order only

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

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

- **first order Markov chain:**

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

- **second order Markov chain:**

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

- **transition probabilities:**

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

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

Markov Chains

Whiteboard

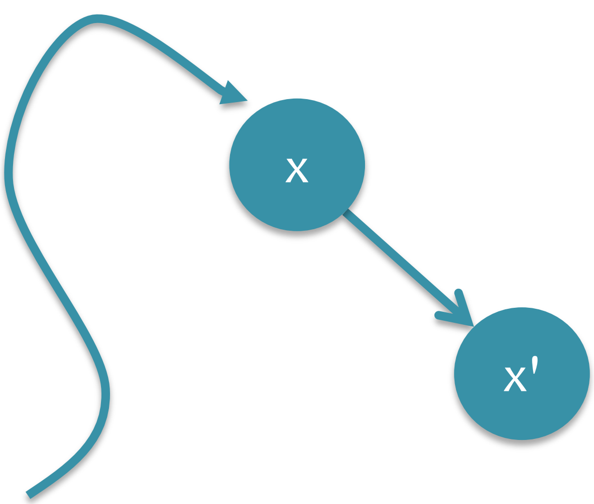
- Invariant distribution
- Equilibrium distribution
- Sufficient conditions for MCMC
- Markov chain as a WFSM

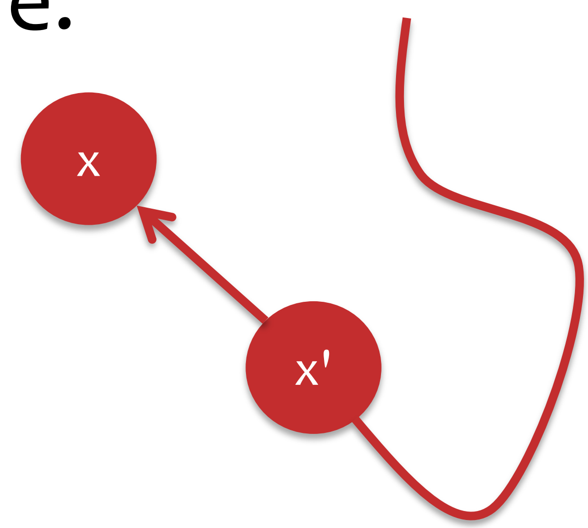
Detailed Balance

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

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 $t \to \infty$
- **Cons**
  - Lots of tunable parameters / design choices
  - Can be quite slow to converge
  - Difficult to tell whether it's working

Extra Slides

Slice Sampling, Hamiltonian Monte Carlo

MCMC (AUXILIARY VARIABLE METHODS)

Auxiliary variables

Extra Slides

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

$$\int f(x)P(x)\,\mathrm{d}x = \int f(x)P(x,v)\,\mathrm{d}x\,\mathrm{d}v$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad x,v \sim P(x,v)$$

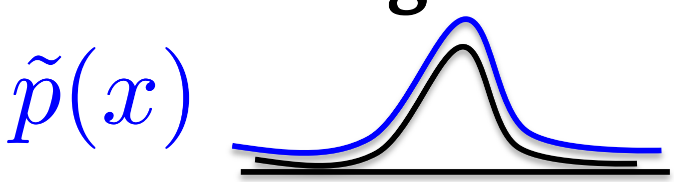
We might want to do this if

- $P(x|v)$ 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**

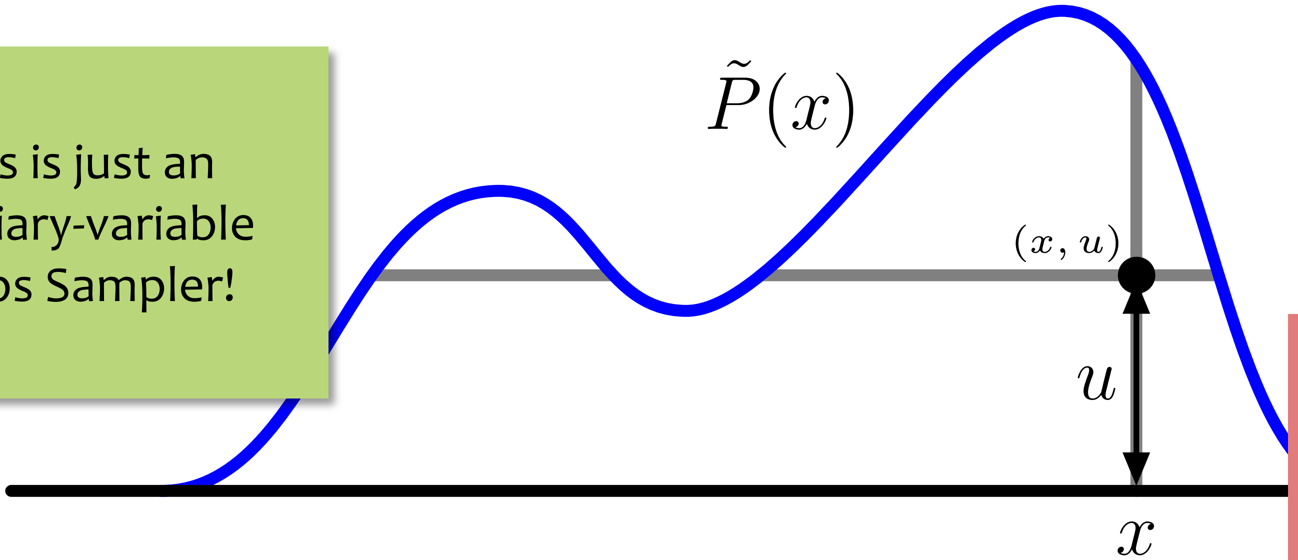


  - An MCMC method with an **adaptive proposal**

Extra Slides

Slice sampling idea

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



This is just an auxiliary-variable Gibbs Sampler!

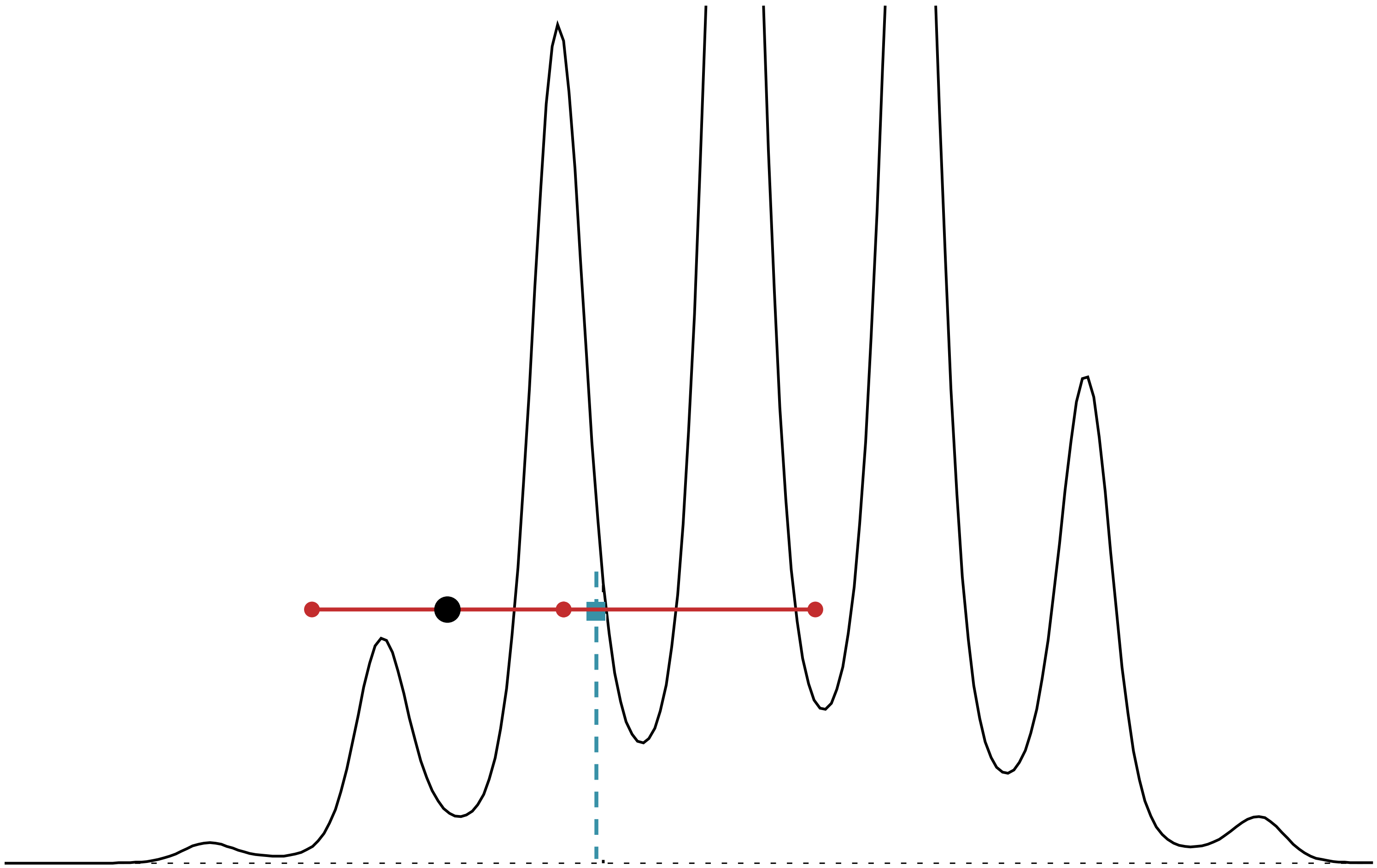
Problem: Sampling from the conditional $p(x \mid u)$ might be infeasible.

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

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

Extra Slides

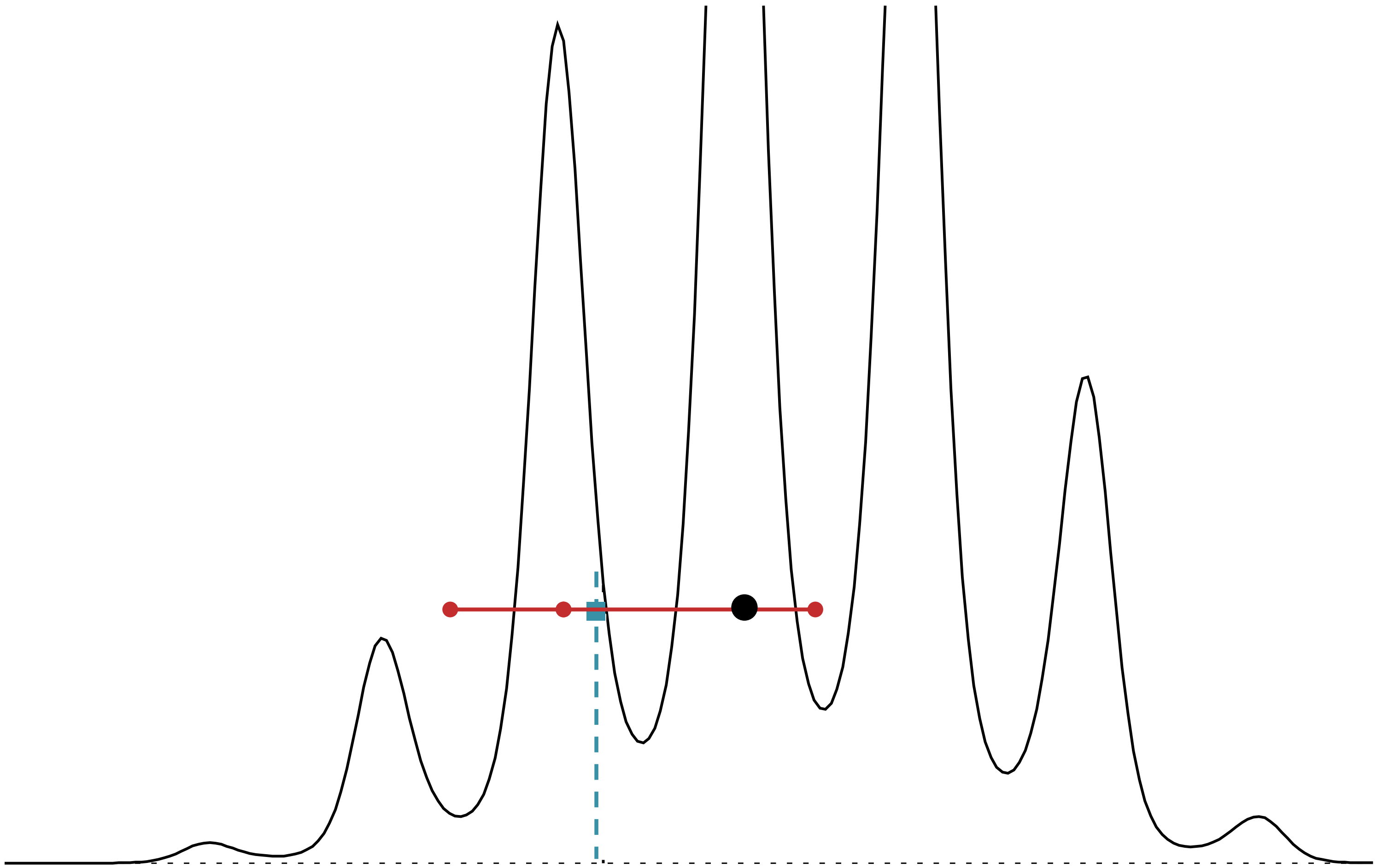
Slice Sampling



ted from MacKay Ch. 29

Extra Slides

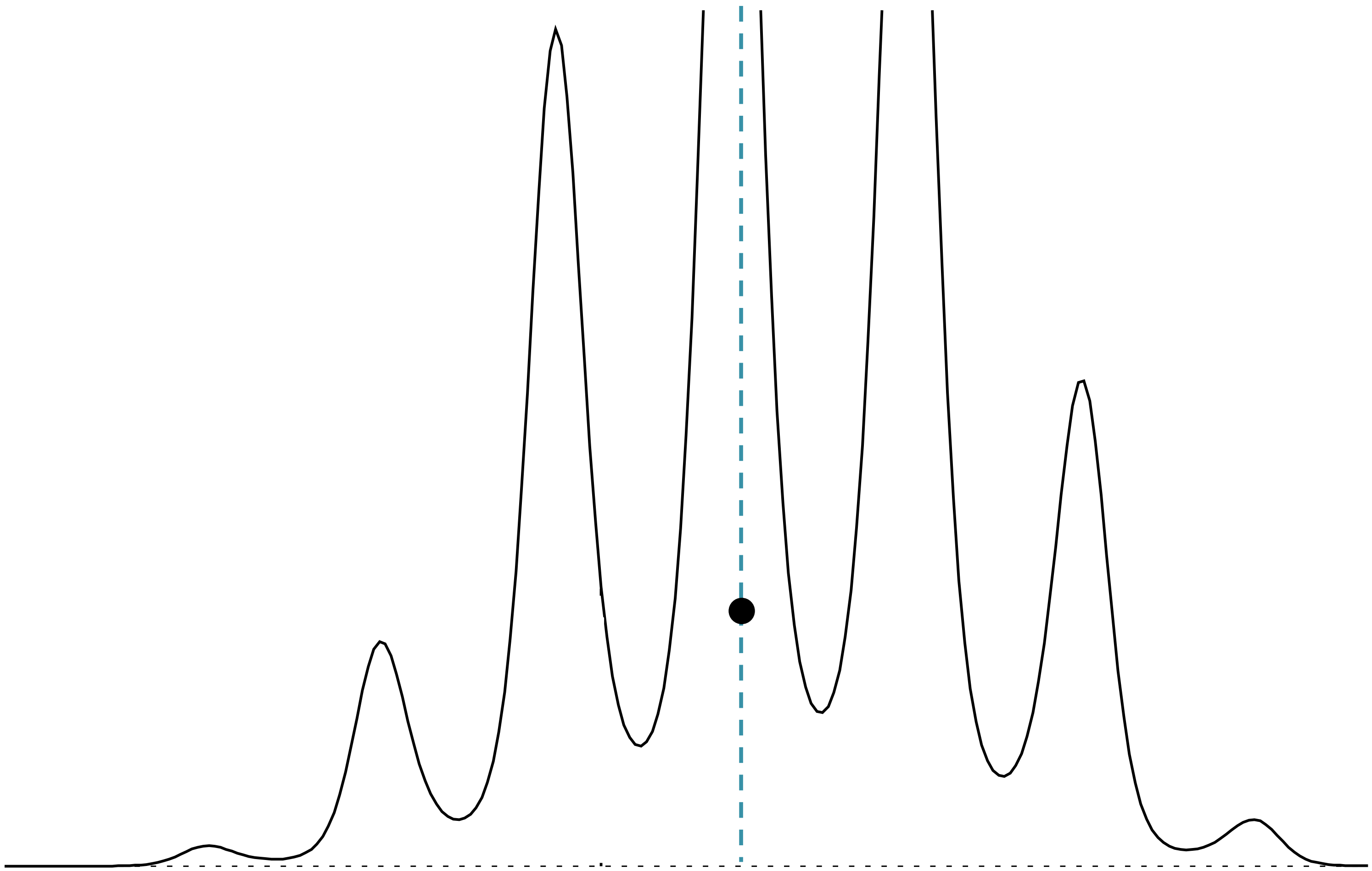
Slice Sampling



ted from MacKay Ch. 29

Extra Slides

Slice Sampling



Slice Sampling

Extra Slides

Algorithm:

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.

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.

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

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

Part 2: Sample x (Shrinking)

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

Slice Sampling

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.

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

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

Part 2: Sample x (Shrinking)

while(true) {

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

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

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

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

$\text{else}\{x_l = x\}$

}

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

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

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

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

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

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

Hamiltonian function

Given a starting position $x^{(1)}$ and a starting momentum $p^{(1)}$ 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, ϵ
2. Number of iterations, L

Leapfrog Algorithm:

$$\text{for } \tau \text{ in } 1 \ldots L\text{:}$$

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

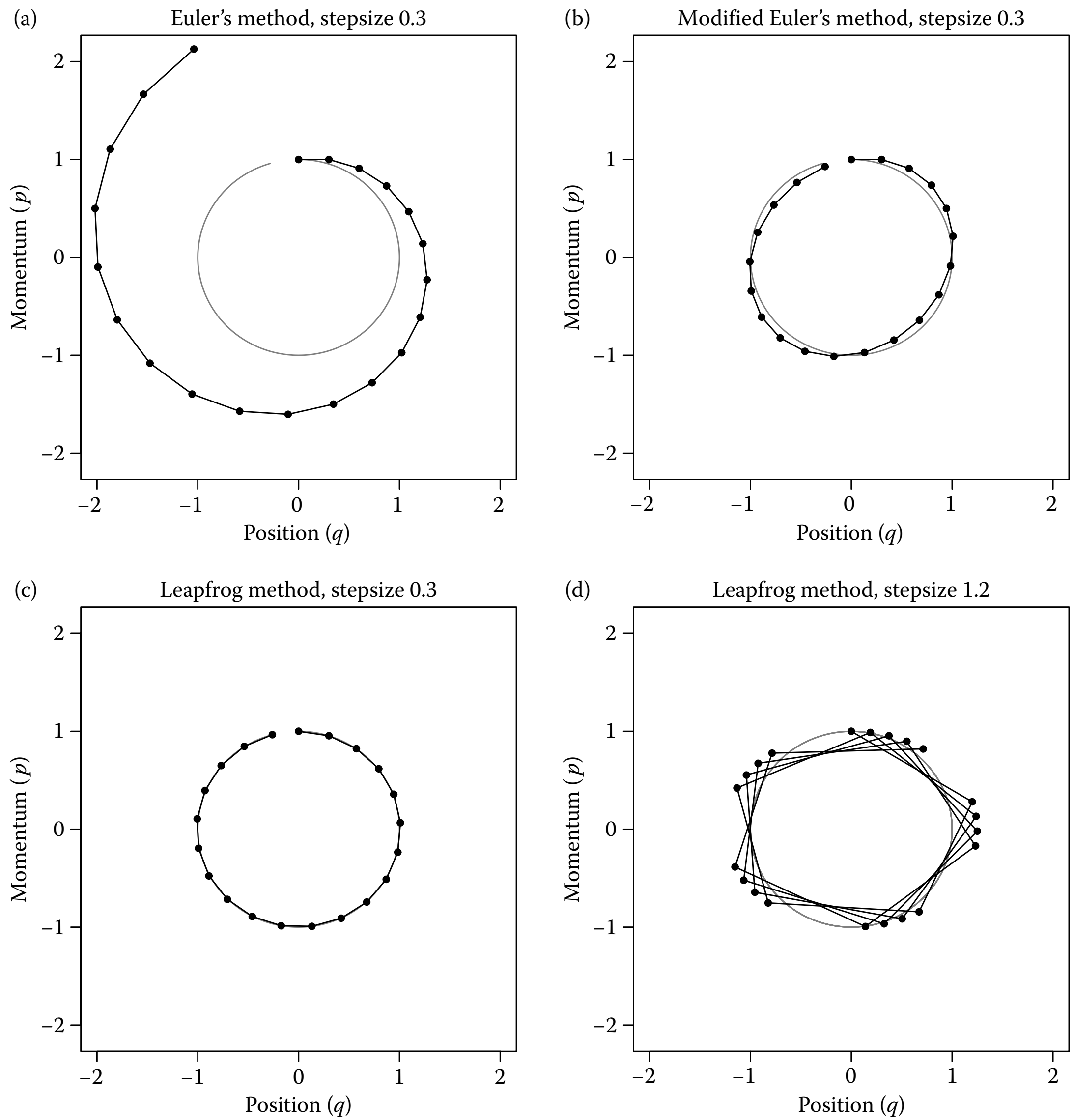
$$\boldsymbol{x} = \boldsymbol{x} + \epsilon \boldsymbol{p}$$

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

from Neal (2011)

Background: Hamiltonian Dynamics

Extra Slides



from Neal (2011)

Extra Slides

Hamiltonian Monte Carlo

Preliminaries

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

Define:

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

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

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

Note:
Since p(x,p) is separable…

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

Target dist.

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

Gaussian

Whiteboard

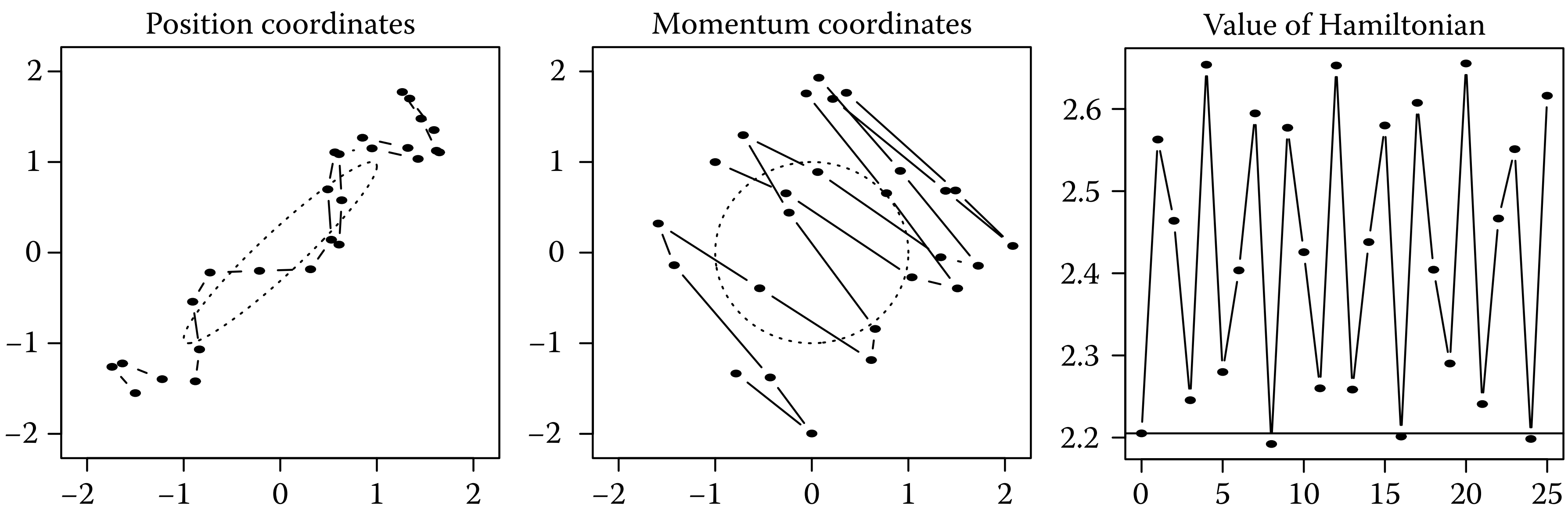
Extra Slides

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

from Neal (2011)

Hamiltonian Monte Carlo

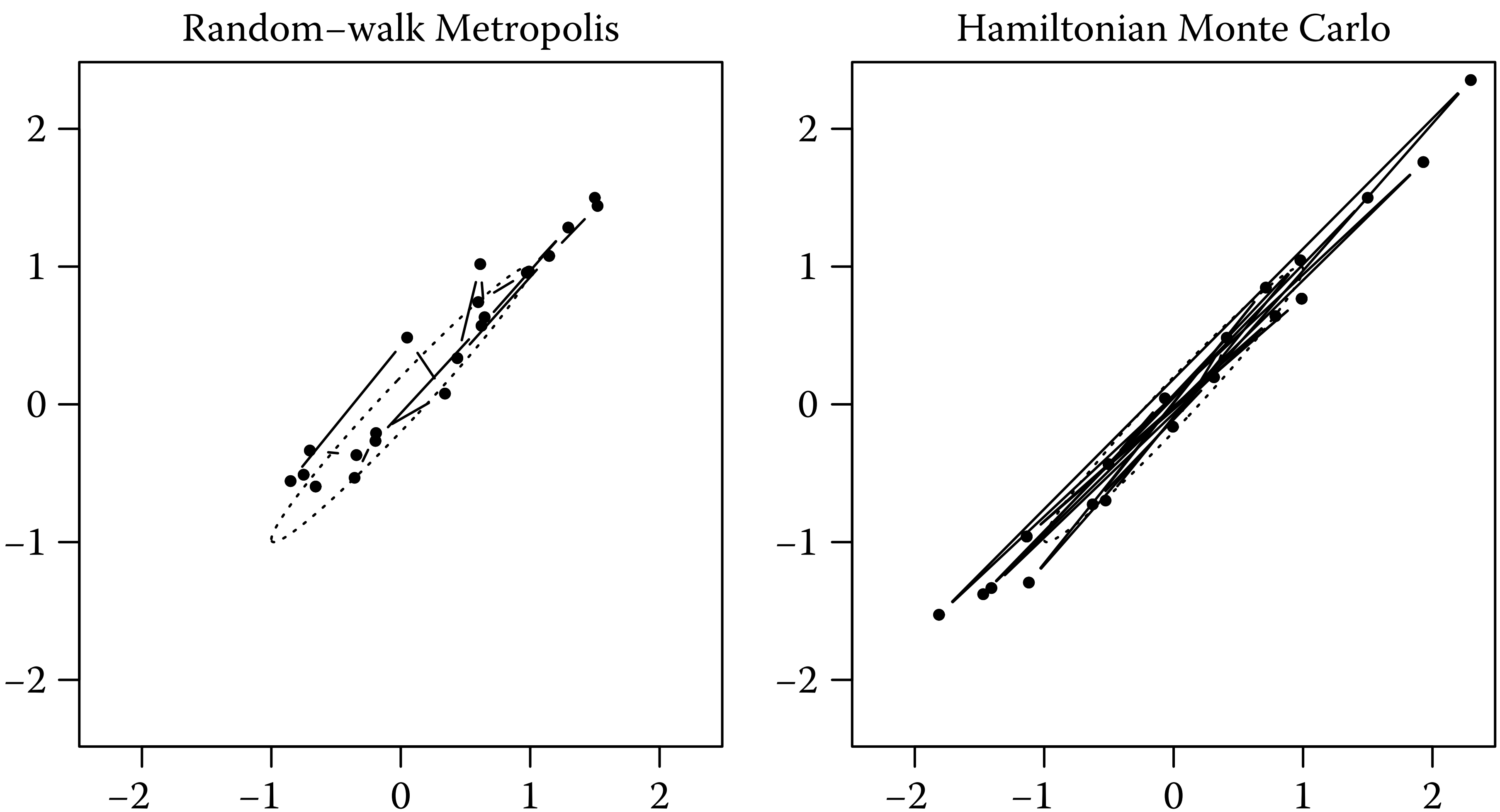
Extra Slides



from Neal (2011)

M-H vs. HMC

Extra Slides



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***

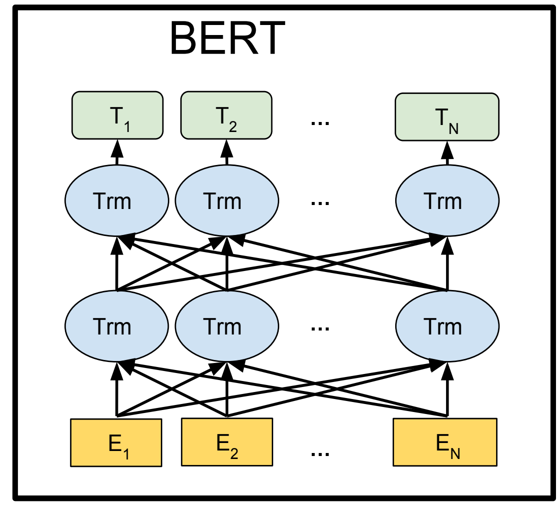
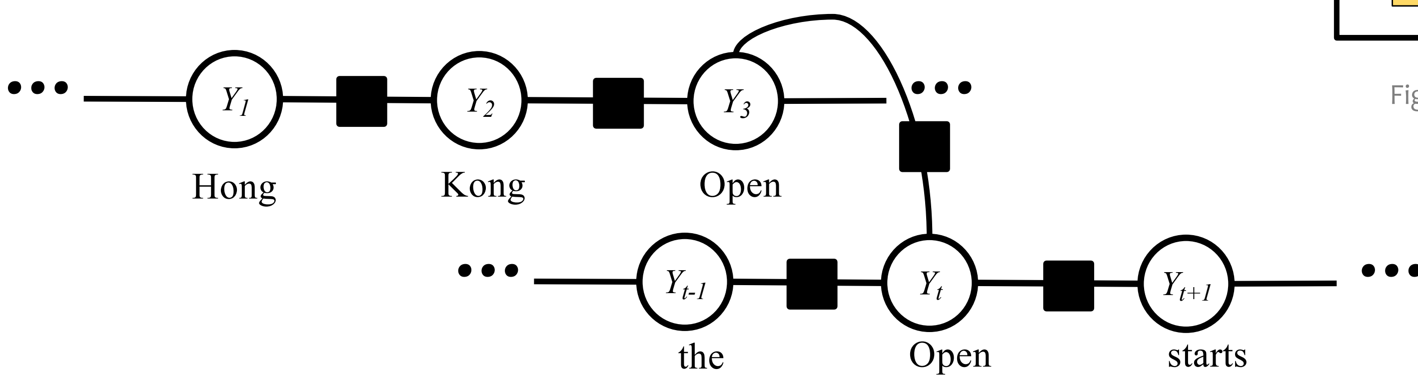
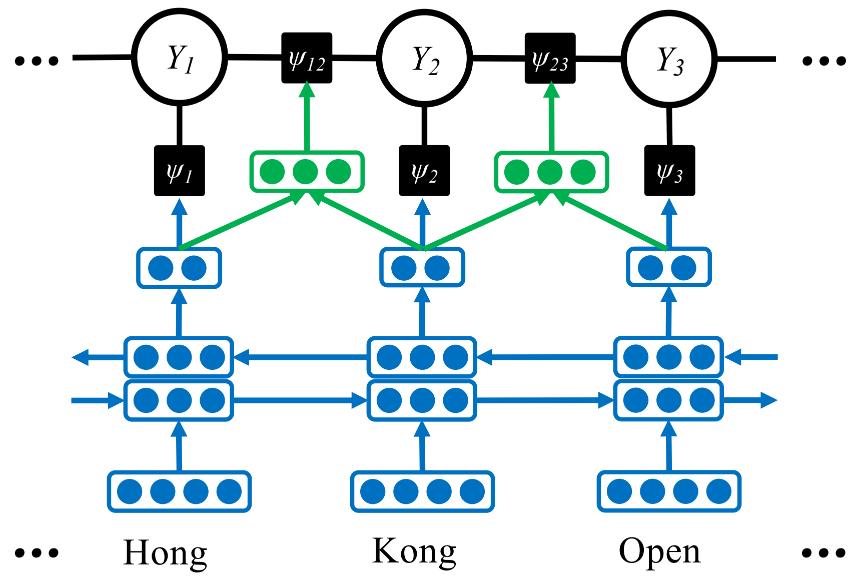


Figure from Devlin et al. (2018)



Background: Linear-chain CRF

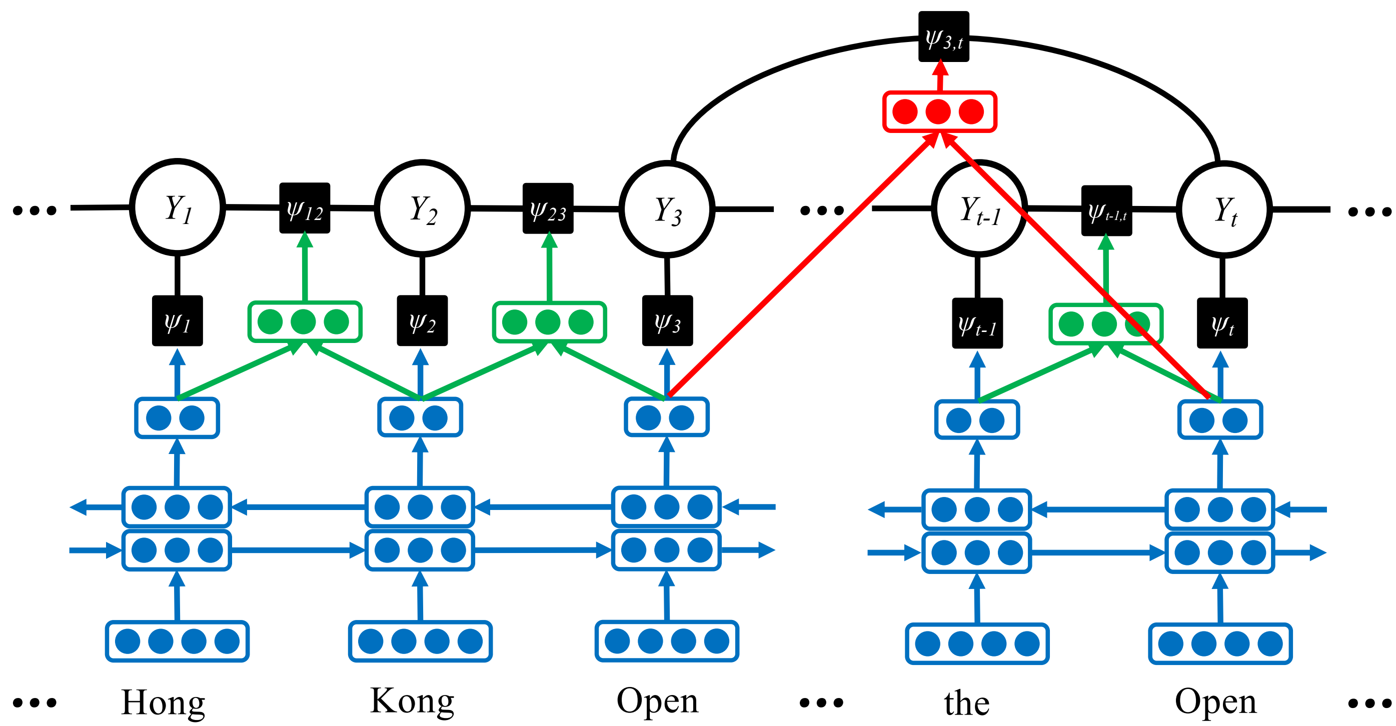
- Prior state-of-the-art approaches for sequence labeling have adopted linear-chain CRFs
  - Model bi-gram dependencies of adjacent labels
  - Exact inference can be done in polynomial time with forward-backward and Viterbi
- We are interested in more complex and expressive CRFs
  - Exact inference may 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:

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

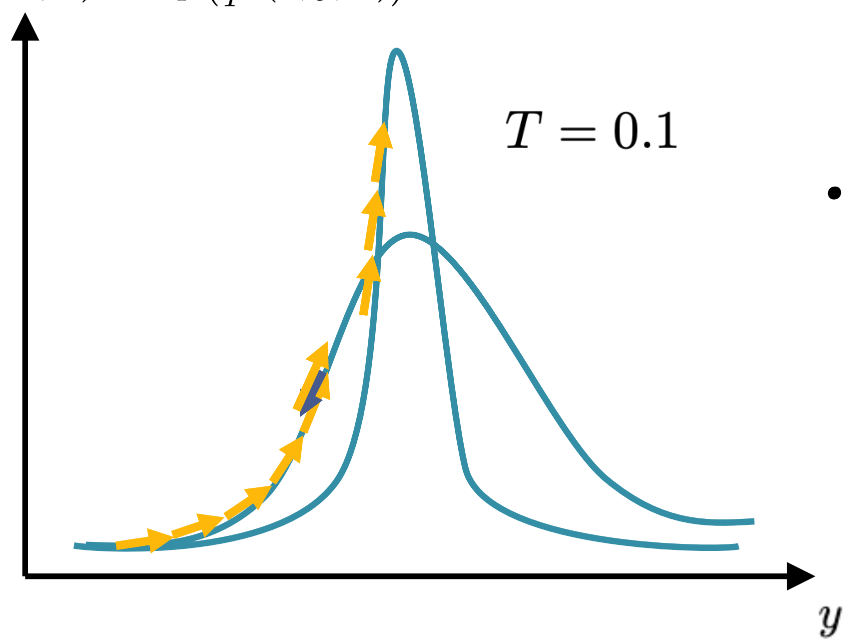
- Training time inference: compute the partition function
- Inference time: find the output with the highest probability

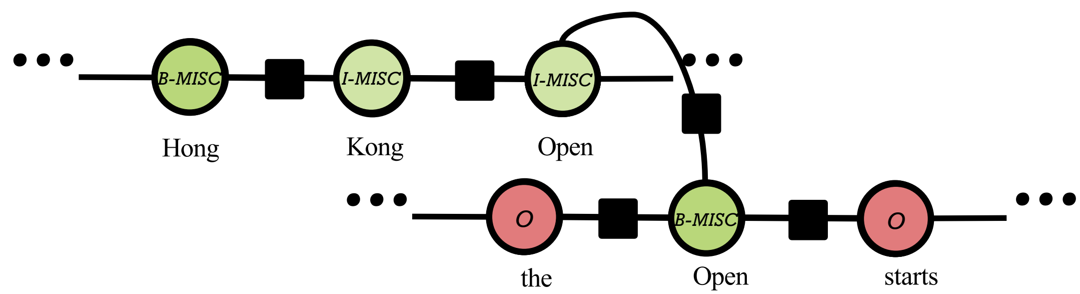
$$\hat{y} = \boxed{\underset{y \in \mathcal{Y}(x)}{\arg\max}}\, s(x,y;\Theta)$$

Inference for Neural CRFs

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

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



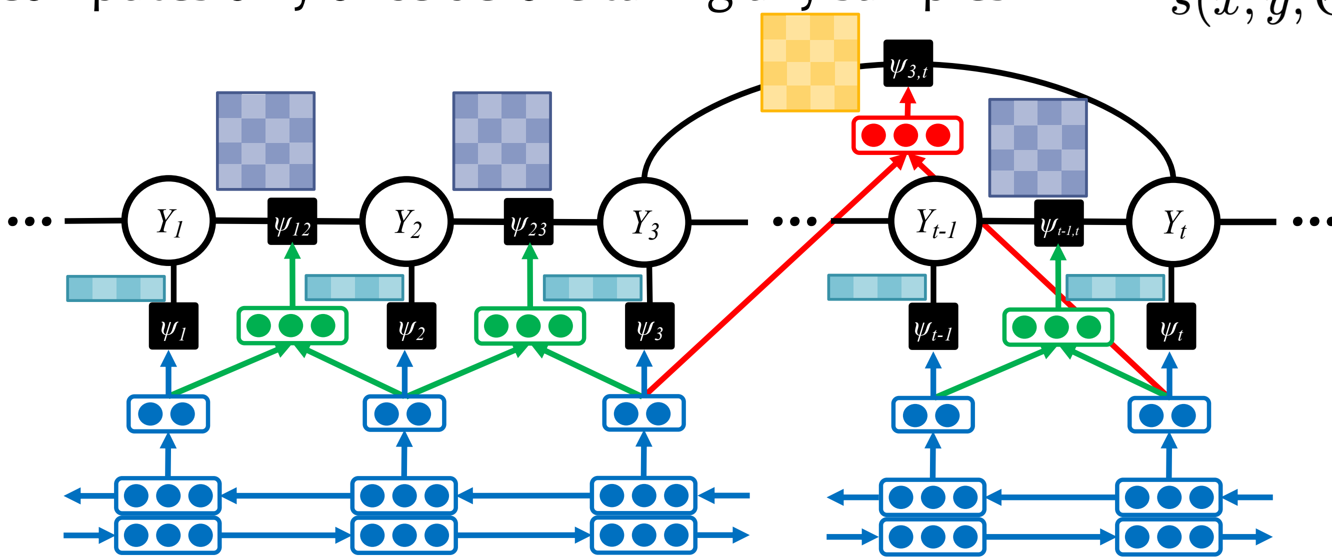


Computational Efficiency

- 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

$$\boxed{z = f(x; \theta_N)}$$

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

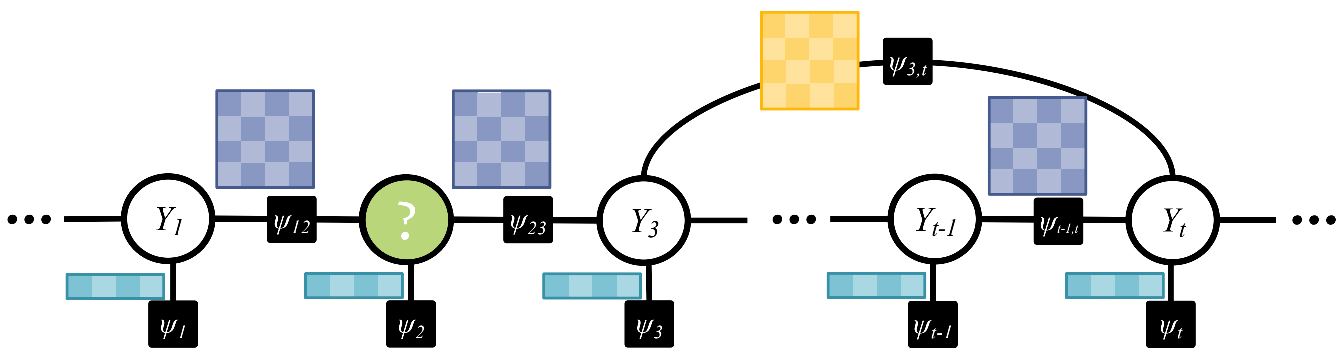


Computational Efficiency

- Decompose the scoring function for computational efficiency
- Graphical model component:
  - Depends on both input and output, but cheap to compute
  - Local computation to take each sample

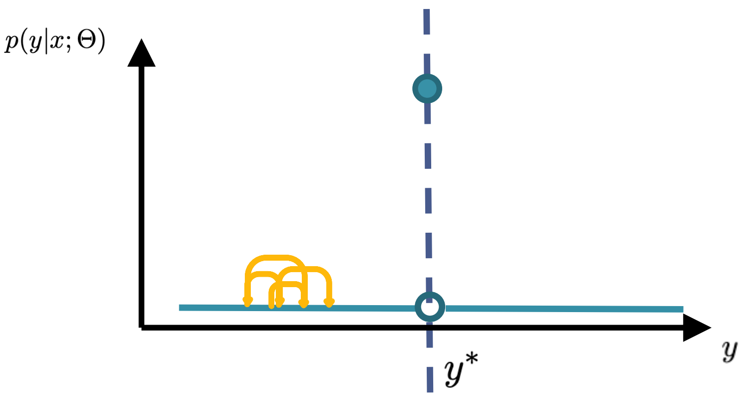
$$z = f(x; \theta_N)$$

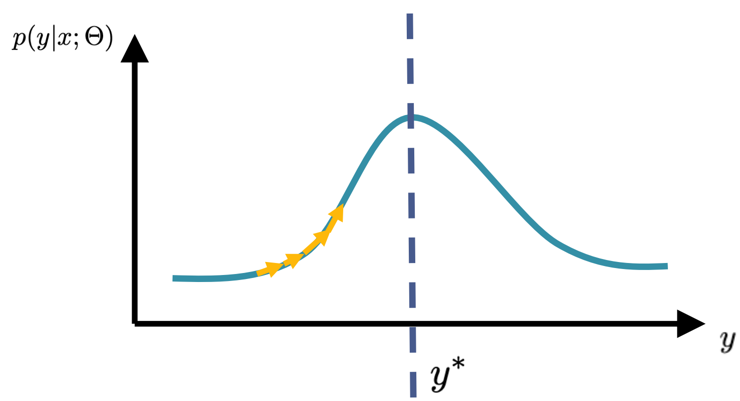
$$\boxed{s(x, y; \Theta) = s\big(x, y, z; \theta_G\big)}$$



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

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

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

Metric (e.g.
negative hamming

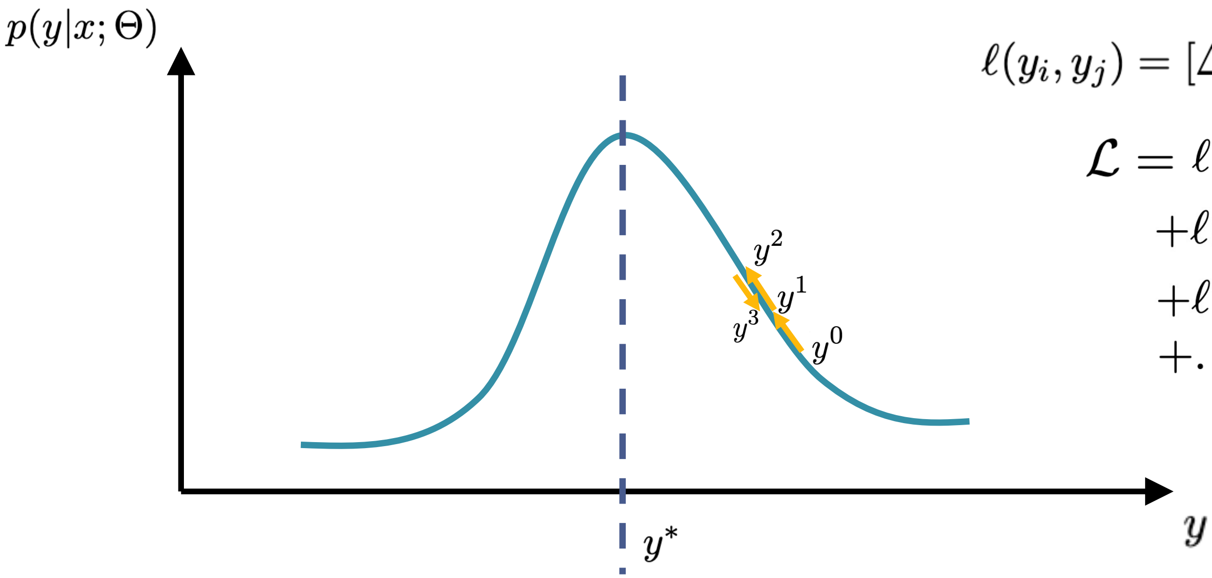
$$y^* = \arg\max\nolimits_{y \in \mathcal{Y}} \omega(y)$$

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

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

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



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

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

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 er al., 2019)	MLE	**93.18**	88.27	90.44
Our baseline Flair	MLE	92.58	**88.30**	90.63
+ skip-chain CRF	NSR	92.56	87.97	**91.44***

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

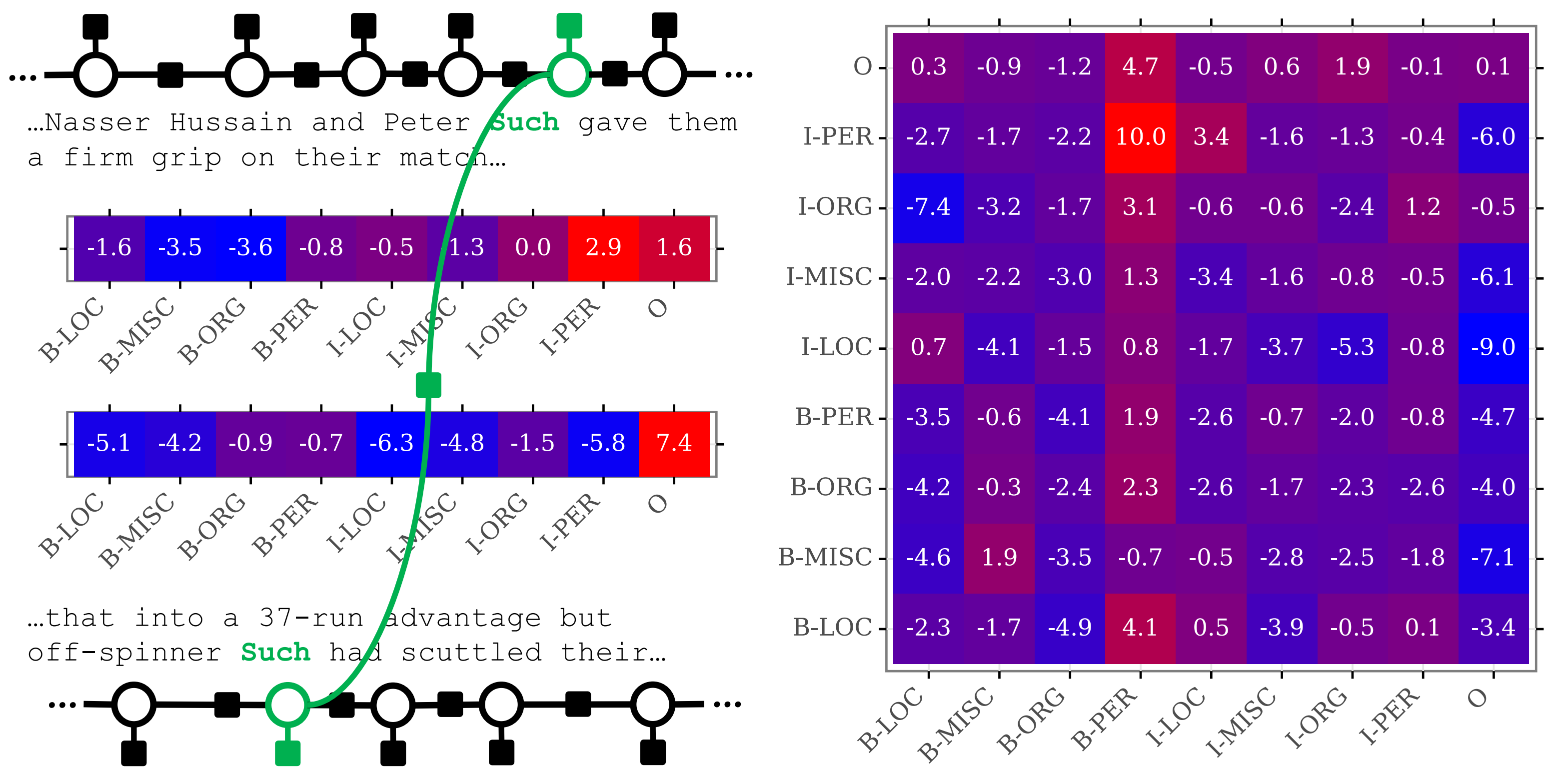
Results: NER (CoNLL-02/03)

- 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	**91.68***

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	**84.50***

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 there's an even easier way…

SUPERVISED LEARNING FOR BAYES NETS (BY "COUNTING")

Recipe for Closed-form MLE

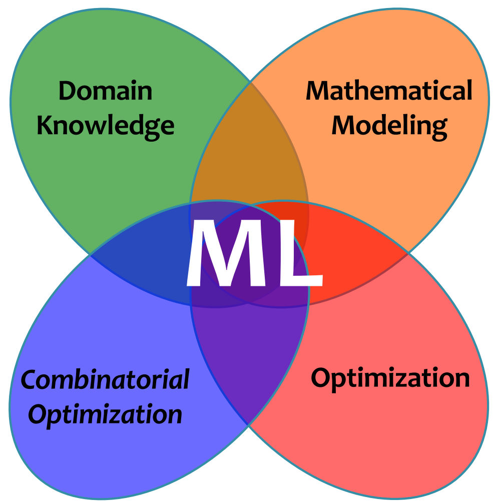
1. Assume data was generated i.i.d. from some model (i.e. write the generative story)
   $x^{(i)} \sim p(x|\boldsymbol{\theta})$
2. Write log-likelihood
   $\ell(\boldsymbol{\theta}) = \log p(x^{(1)}|\boldsymbol{\theta}) + \ldots + \log p(x^{(N)}|\boldsymbol{\theta})$
3. Compute partial derivatives (i.e.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

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



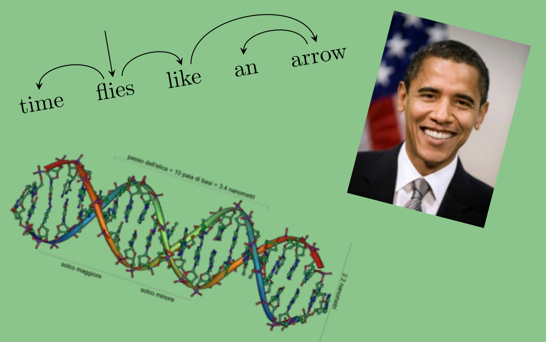
Inference finds {best structure, marginals, partition function} for a new observation

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

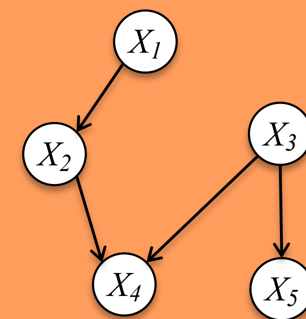
Learning tunes the parameters of the model

Machine Learning

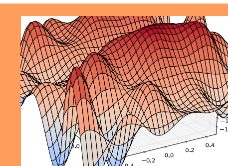
Data



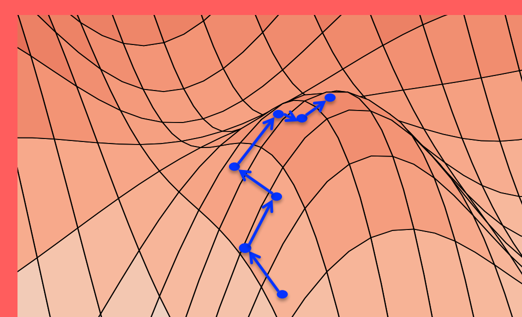
Model



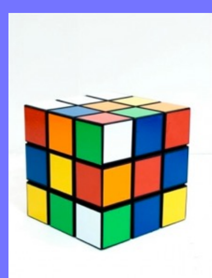
Objective



Learning

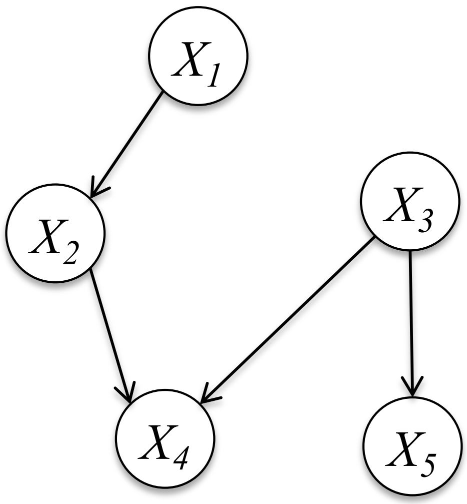


Inference



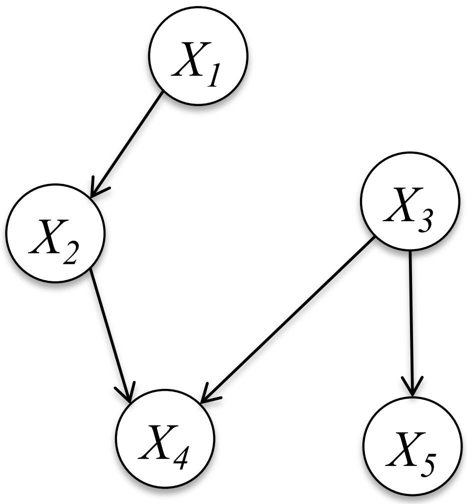
(**Inference** is usually called as a subroutine in learning)

Learning Fully Observed BNs



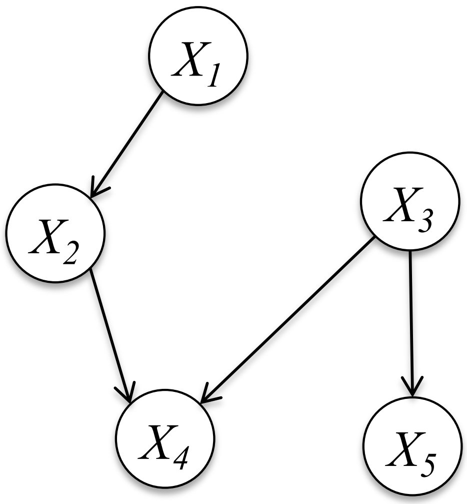
$$
\begin{aligned}
p(X_1, X_2, X_3, X_4, X_5) = \\
p(X_5|X_3)p(X_4|X_2, X_3) \\
p(X_3)p(X_2|X_1)p(X_1)
\end{aligned}
$$

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



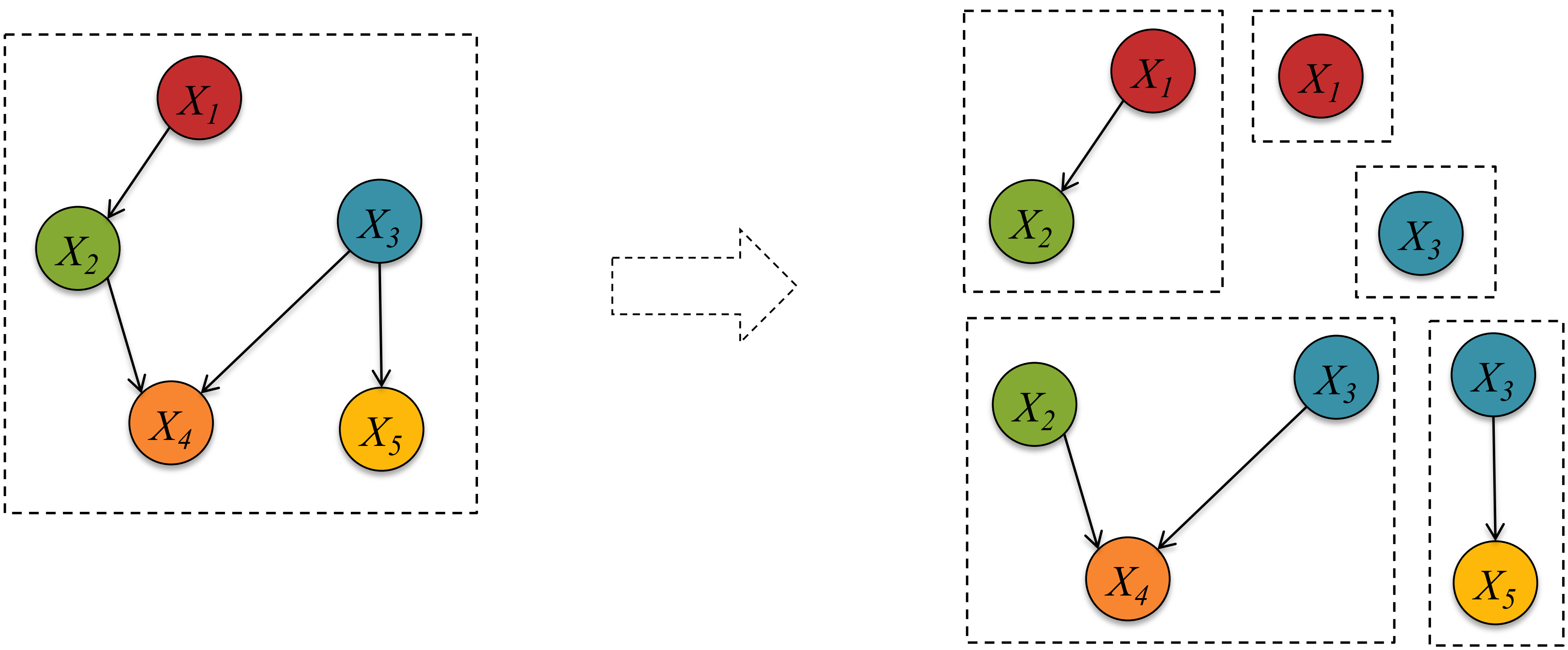
$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$\color{red}{p(X_5|X_3)p(X_4|X_2, X_3)}$$
$$\color{blue}{p(X_3)}\color{red}{p(X_2|X_1)}\color{blue}{p(X_1)}$$

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

Learning Fully Observed BNs

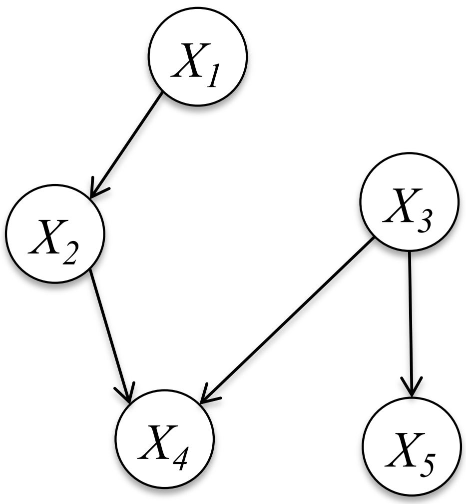
Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data

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



Learning Fully Observed BNs

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



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

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

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

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

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

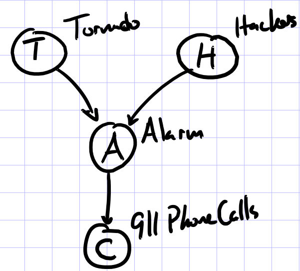
$$\theta_5^* = \operatorname*{argmax}_{\theta_5} \log p(X_5|X_3, \theta_5)$$

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



$H \sim \text{Bernoulli}(\eta)$
$T \sim \text{Bernoulli}(\tau)$
$A \sim \text{Bernoulli}(\alpha_{H,T})$
$C \sim \text{Uniform}(\{1,\ldots,6\}) + A * \text{Uniform}(\{1,\ldots,6\})$

parameters

no parameters

integer

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

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

MLEs in Closed Form

$$\ell(\eta,\tau,\alpha) = \log \prod_{i=1}^{12} p(t^{(i)}, h^{(i)}, a^{(i)}, c^{(i)} | \eta, \tau, \alpha)$$

$$= \sum_{i=1}^{12} \log p(t^{(i)}|\tau) + \log p(h^{(i)}|\eta) + \log p(a^{(i)}|t^{(i)}, h^{(i)}, \alpha) + \log p(c^{(i)}|a^{(i)})$$

$$\hat{\eta}, \hat{\tau}, \hat{\alpha} = \operatorname{argmax} \ell(\eta,\tau,\alpha)$$

$$\hat{\eta} = \operatorname*{argmax}_{\eta} \sum_{i=1}^{12} \log p(h^{(i)}|\eta) = \#(T=1)/N$$

$$\hat{\tau} = \operatorname*{argmax}_{\tau} \sum_{i=1}^{12} \log p(t^{(i)}|\tau) = \#(H=1)/N$$

$$\hat{\alpha} = \operatorname*{argmax}_{\alpha} \sum_{i=1}^{12} \log p(a^{(i)}|t^{(i)}, h^{(i)}, \alpha)$$

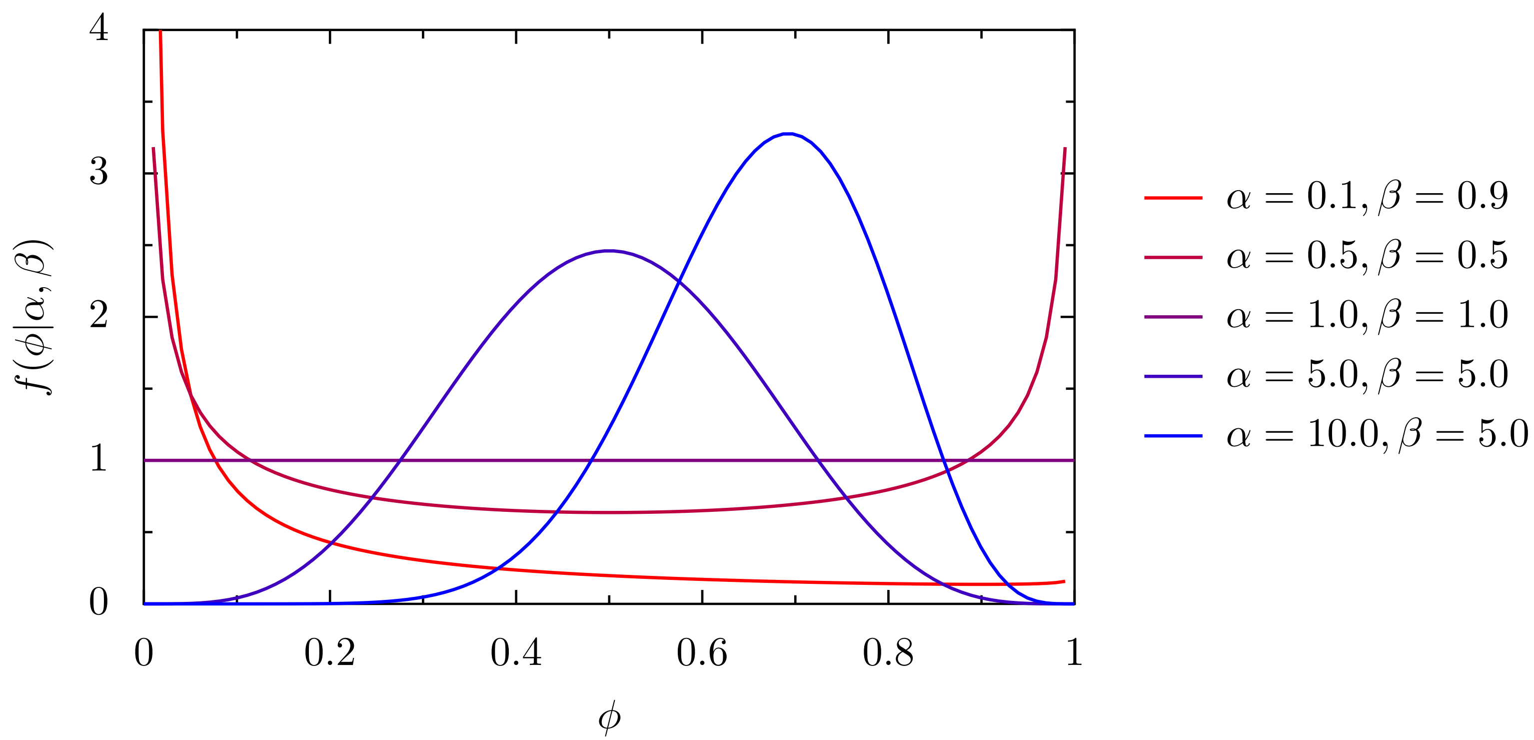
$$\hat{\alpha}_{t,h} = \frac{\#(A=1, T=t, H=h)}{\#(T=t, H=h)}$$

BAYESIAN INFERENCE FOR NAÏVE BAYES

Beta-Bernoulli Model

- Beta Distribution

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



Beta-Bernoulli Model

- Generative Process

$\phi \sim \text{Beta}(\alpha, \beta)$	[*draw distribution over words*]
For each word $n \in \{1, \ldots, N\}$	
$\quad x_n \sim \text{Bernoulli}(\phi)$	[*draw word*]

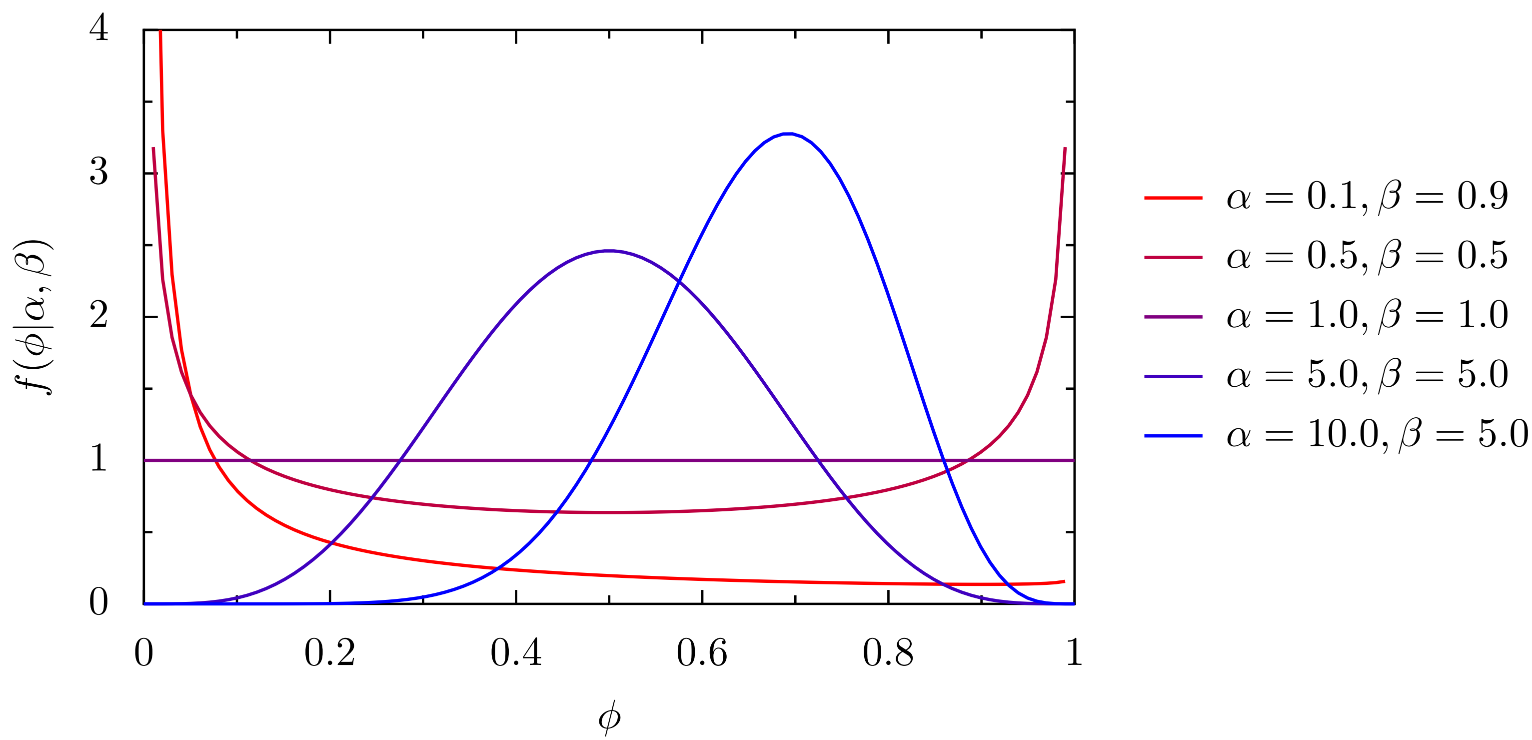
- Example corpus (heads/tails)

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

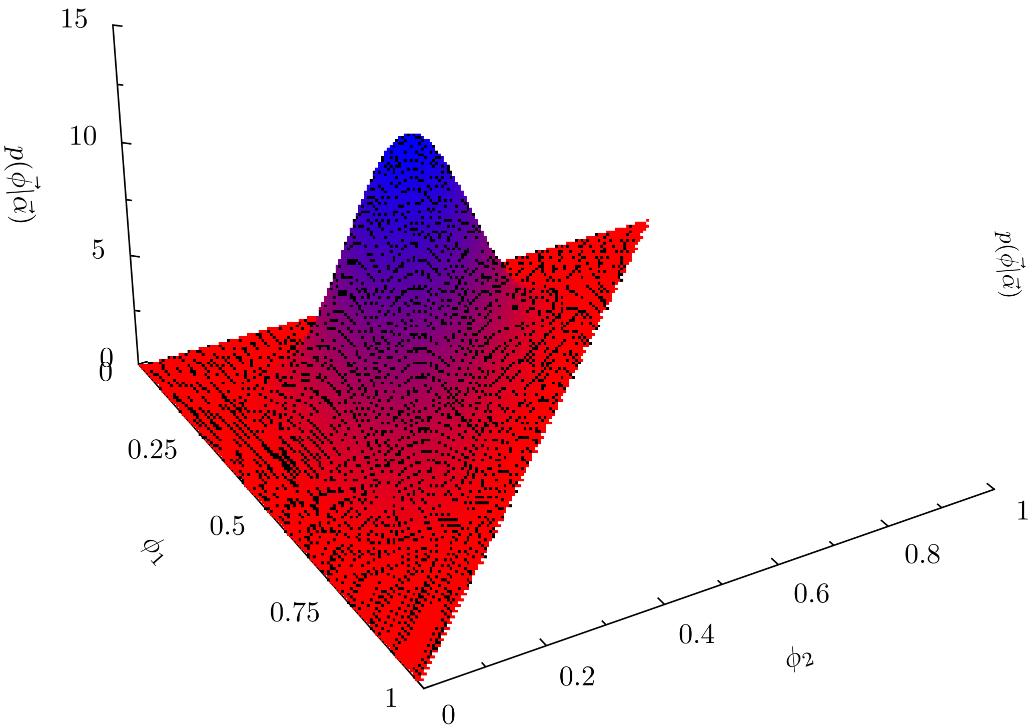
$$f(\phi|\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

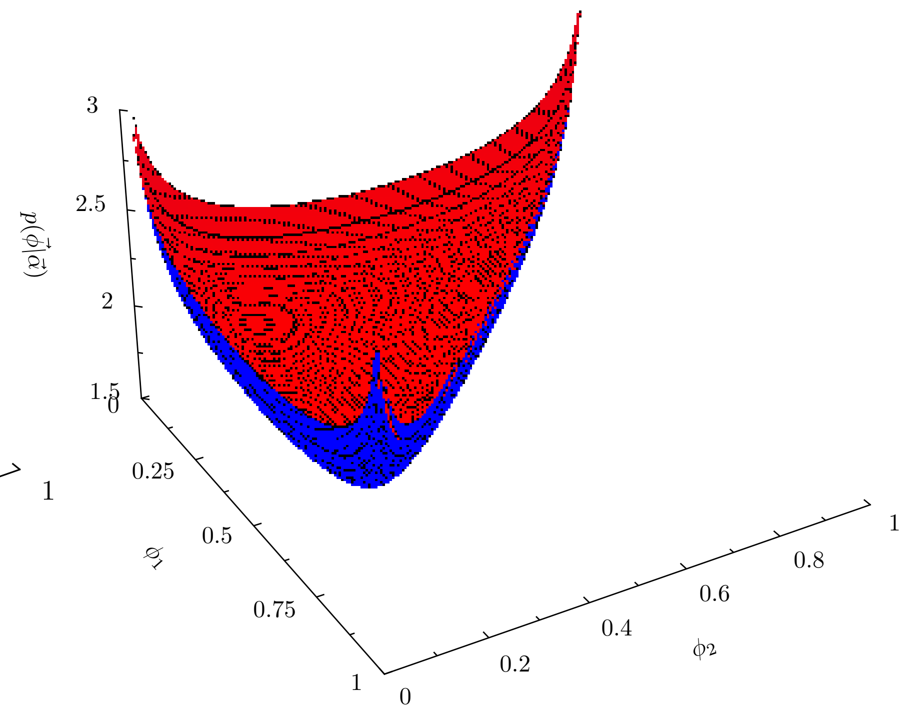


Dirichlet-Multinomial Model

- Dirichlet Distribution

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





Dirichlet-Multinomial Model

- Generative Process

$$\begin{array}{ll} \boldsymbol{\phi} \sim \text{Dir}(\boldsymbol{\beta}) & [\textit{draw distribution over words}] \\ \text{For each word } n \in \{1, \ldots, N\} & \\ \quad x_n \sim \text{Mult}(1, \boldsymbol{\phi}) & [\textit{draw word}] \end{array}$$

- 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

$\boldsymbol{\phi} \sim \text{Dir}(\boldsymbol{\beta})$	[*draw distribution over words*]
For each word $n \in \{1, \ldots, N\}$	
$\quad x_n \sim \text{Mult}(1, \boldsymbol{\phi})$	[*draw word*]

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

$$p(\vec{\phi} | \vec{x}, \vec{\beta}) \propto p(\vec{x}|\vec{\phi})\, p(\vec{\phi})$$

$$= \left[\prod_{i=1}^{N} p(x^{(i)}|\vec{\phi})\right] p(\vec{\phi})$$

$$\propto \left[\prod_{k=1}^{K} \phi_k^{\beta_k - 1}\right] \left[\prod_{i=1}^{N} \prod_{k=1}^{K} \phi_k^{\mathbb{1}(x^{(i)}=k)}\right]$$

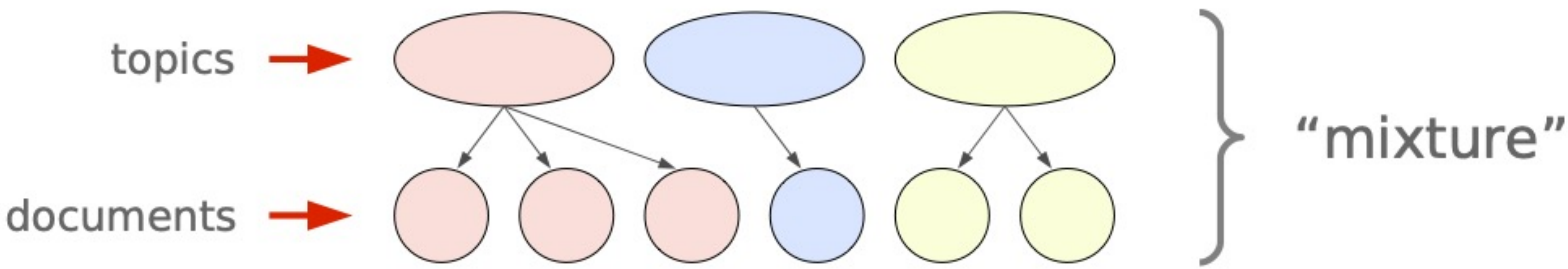
$$= \prod_{k=1}^{K} \phi_k^{\left[\beta_k - 1 + \sum_{i=1}^{N} \mathbb{1}(x^{(i)}=k)\right]}$$

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

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

Figure from Wallach, JHU 2011, slides

Dirichlet-Multinomial Mixture Model

- Generative Process

For each topic $k \in \{1, \ldots, K\}$:
    $\boldsymbol{\phi}_k \sim \text{Dir}(\boldsymbol{\beta})$ [*draw distribution over words*]
$\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$ [*draw distribution over topics*]
For each document $m \in \{1, \ldots, M\}$
    $z_m \sim \text{Mult}(1, \boldsymbol{\theta})$ [*draw topic assignment*]
    For each word $n \in \{1, \ldots, N_m\}$
        $x_{mn} \sim \text{Mult}(1, \boldsymbol{\phi}_{z_m})$ [*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