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

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Conditional Independencies
+
Factor Graphs

Matt Gormley
Lecture 7
Sep. 23, 2022

Reminders

- **Homework 2: Learning to Search for RNNs**
  - Out: Sun, Sep 18
  - Due: Thu, Sep 29 at 11:59pm

Q&A

Q: What is the difference between a stochastic policy and a deterministic policy?

A: Definition: a **stochastic policy** is a probability distribution over actions given a state

$$a_t \sim \pi(\cdot \mid s_t)$$

Definition: a **deterministic policy** is a function that maps from a state to an action

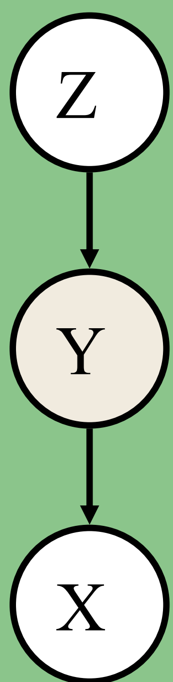
$$a_t = \pi(s_t)$$

CONDITIONAL INDEPENDENCIES OF DGMS

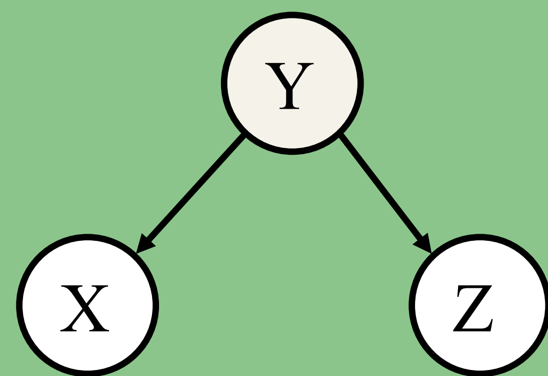
What Independencies does a Bayes Net Model?

Three cases of interest…

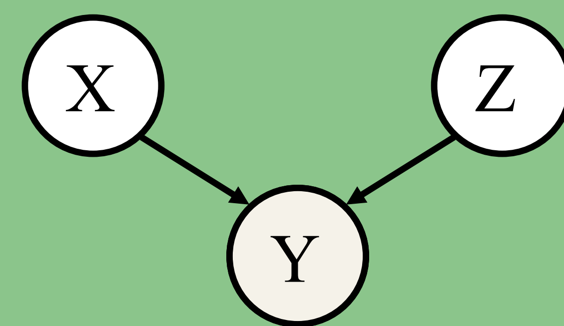
Cascade



Common Parent



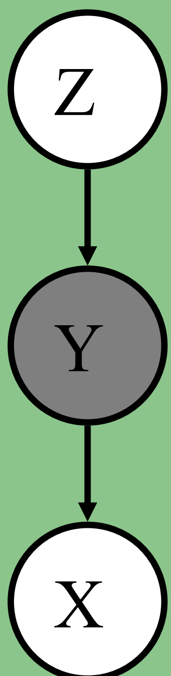
V-Structure



What Independencies does a Bayes Net Model?

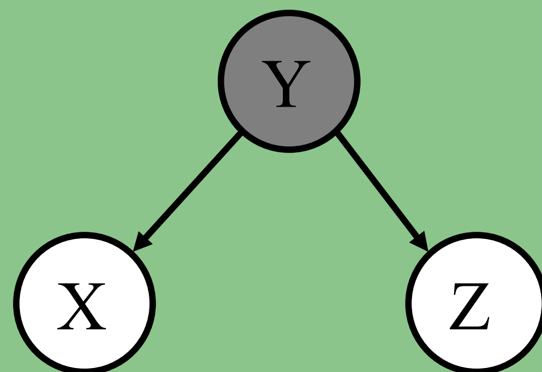
Three cases of interest…

Cascade



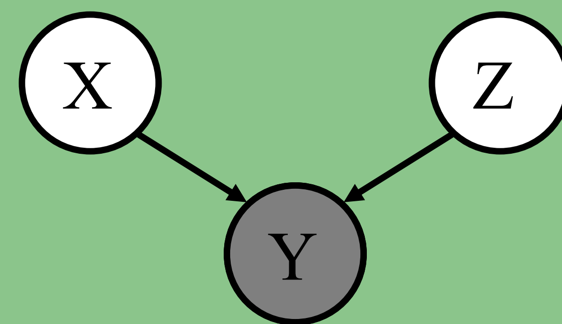
$$X \perp\!\!\!\perp Z \mid Y$$

Common Parent



$$X \perp\!\!\!\perp Z \mid Y$$

V-Structure



$$X \not\perp\!\!\!\perp Z \mid Y$$

Knowing Y
decouples X and Z

Knowing Y
couples X and Z

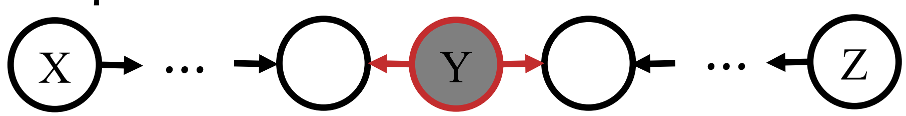
D-Separation

If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

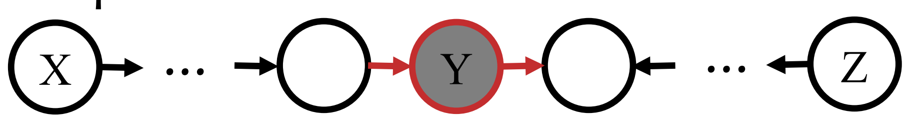
Definition #1:
Variables X and Z are **d-separated** given a **set** of evidence variables E iff every path from X to Z is "blocked".

A path is "blocked" whenever:

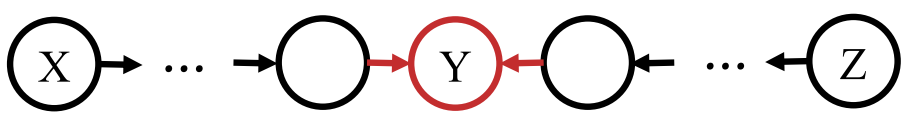
1. $\exists$Y on path s.t. Y $\in$ E and Y is a "common parent"



2. $\exists$Y on path s.t. Y $\in$ E and Y is in a "cascade"



3. $\exists$Y on path s.t. {Y, descendants(Y)} $\notin$ E and Y is in a "v-structure"



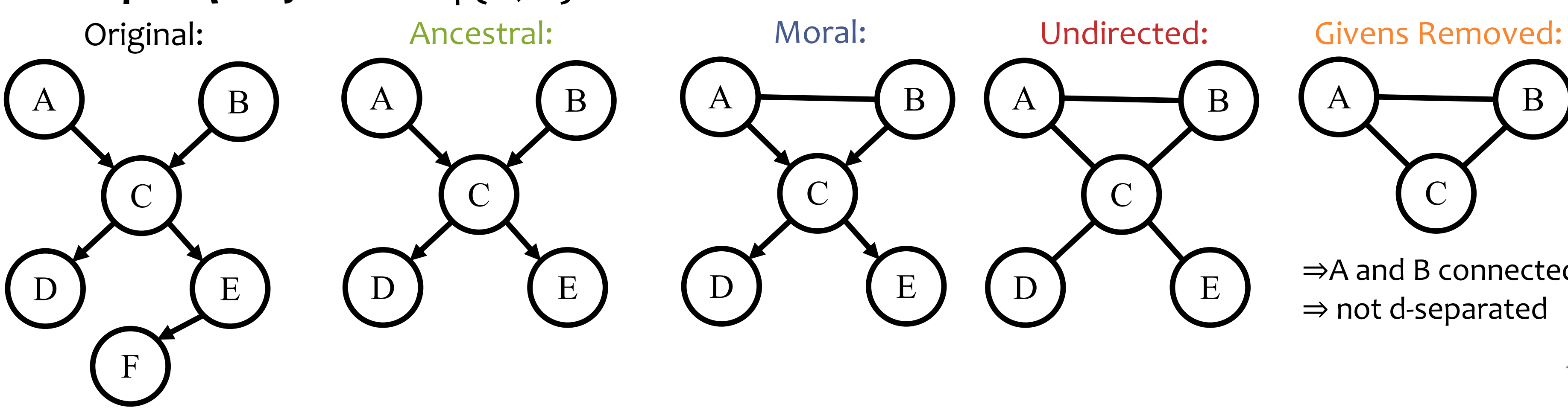
D-Separation

If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

Definition #2:
Variables X and Z are **d-separated** given a **set** of evidence variables E iff there does **not** exist a path in the **undirected ancestral moral** graph **with E removed**.

1. **Ancestral graph**: keep only X, Z, E and their ancestors
2. **Moral graph**: add undirected edge between all pairs of each node's parents
3. **Undirected graph**: convert all directed edges to undirected
4. Givens Removed: delete any nodes in E

Example Query: $A \perp\!\!\!\perp B \mid \{D, E\}$

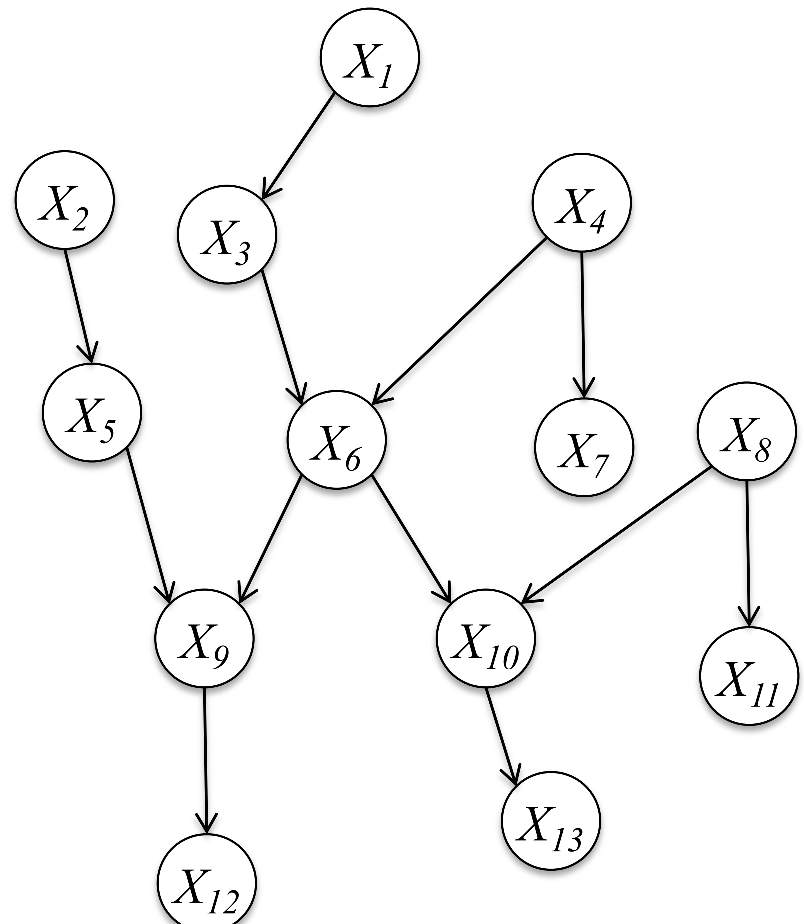


⇒A and B connected
⇒ not d-separated

Markov Blanket (Directed)

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov Blanket** of a node in a directed graphical model is the set containing the node's parents, children, and co-parents.

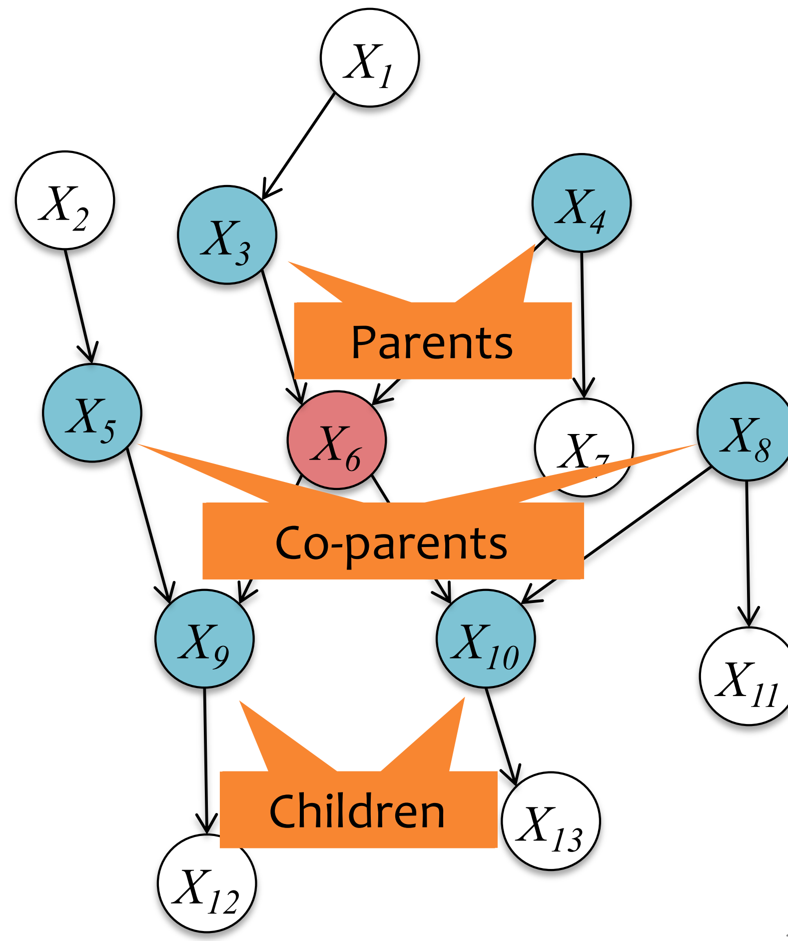


Markov Blanket (Directed)

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov Blanket** of a node in a directed graphical model is the set containing the node's parents, children, and co-parents.

Example: The Markov Blanket of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$



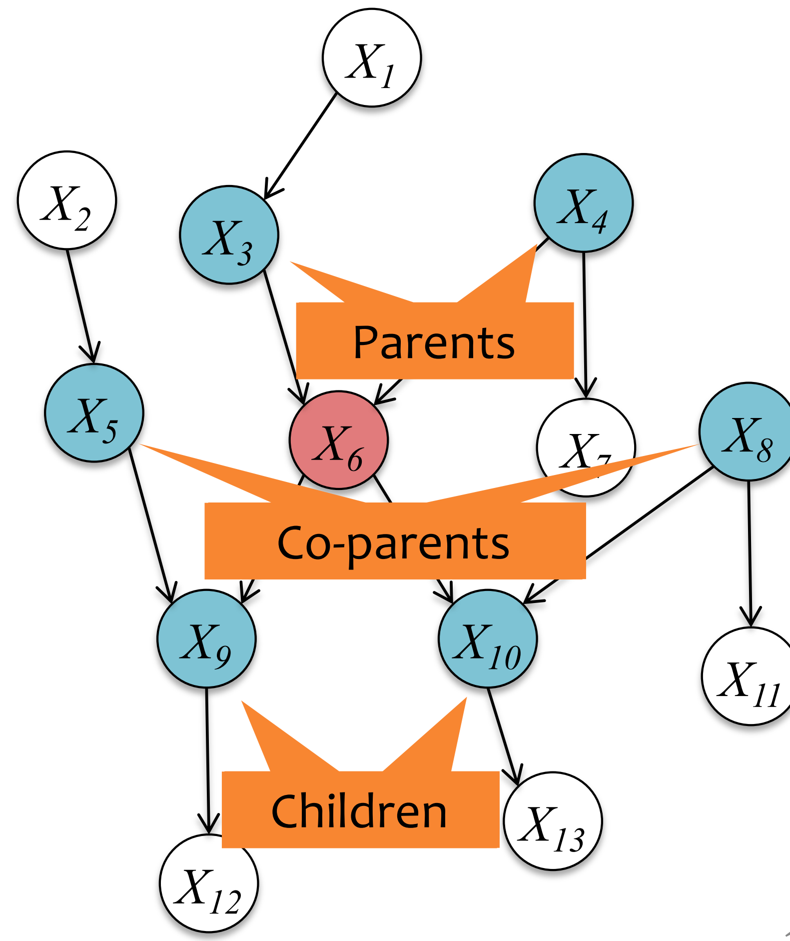
Markov Blanket (Directed)

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov Blanket** of a node in a directed graphical model is the set containing the node's parents, children, and co-parents.

Theorem: a node is **conditionally independent** of every other node in the graph given its **Markov blanket**

Example: The Markov Blanket of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$



CONDITIONAL INDEPENDENCIES OF UGMS

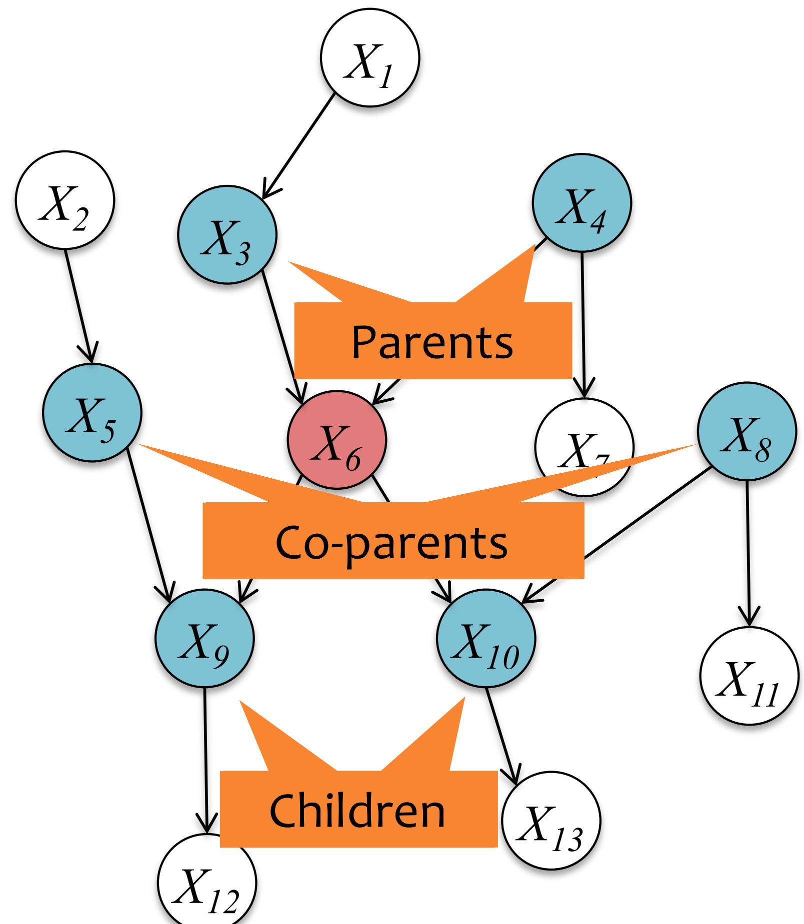
Markov Blanket (Directed)

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov Blanket** of a node in a **directed** graphical model is the set containing the node's parents, children, and co-parents.

Theorem: a node is **conditionally independent** of every other node in the graph given its **Markov blanket**

Example: The Markov Blanket of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$

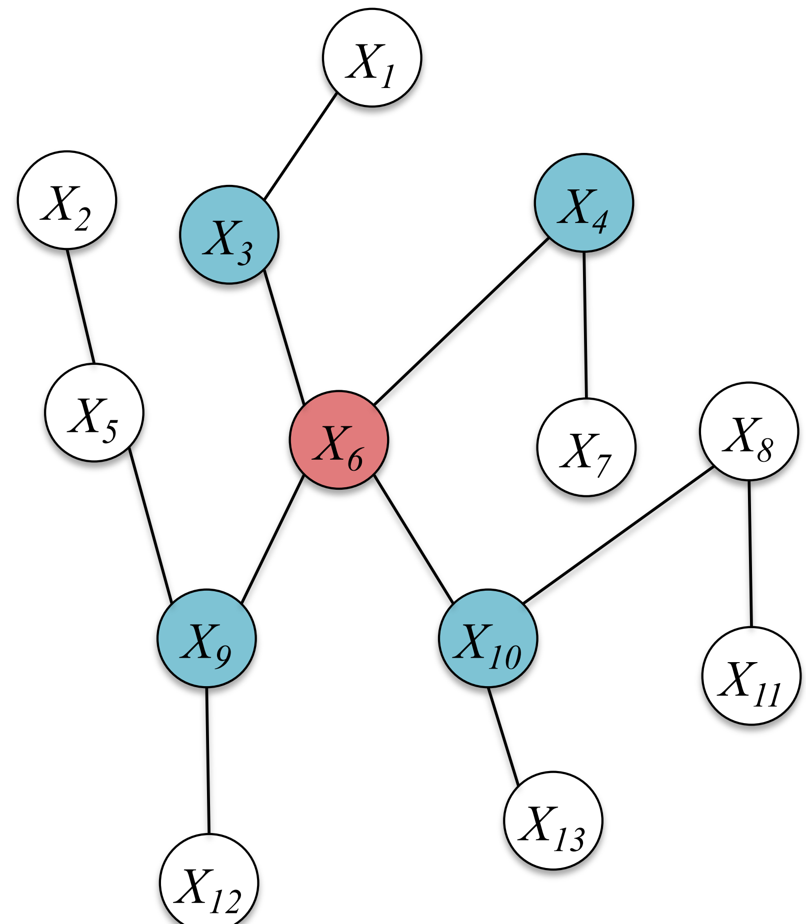


Markov Blanket (Undirected)

Def: the **Markov Blanket** of a node in an **undirected** graphical model is the set containing the node's neighbors.

Theorem: a node is **conditionally independent** of every other node in the graph given its **Markov blanket**

Example: The Markov Blanket of X_6 is $\{X_3, X_4, X_9, X_{10}\}$



Undirected Graphical Models

$A = \{1, 2, 3\}$

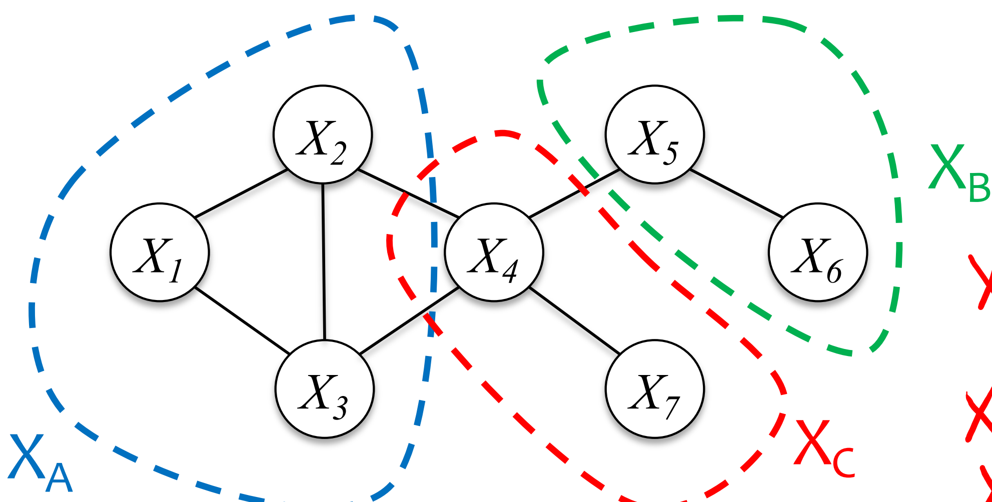
$X_A = \{X_1, X_2, X_3\}$

<u>*Conditional Independence Semantics*</u>

Consider a distribution over r.v.s $X_1, \ldots, X_T$

For a UGM and any disjoint index sets A, B, C,
(i.e., $A \subseteq \{1, \ldots, T\}$, $B \subseteq \{1, \ldots, T\}$, $C \subseteq \{1, \ldots, T\}$)

X_A is **conditionally independent** of X_B given X_C iff
X_C separates sets X_A and X_B



$X_A \perp\!\!\!\perp X_B \mid X_C$

$X_i \perp\!\!\!\perp X_j \mid X_c$

$X_i \in X_A$
$X_j \in X_B$

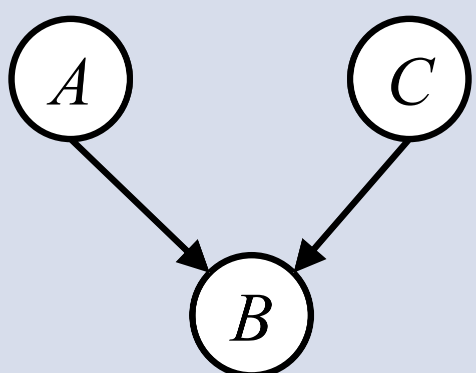
Undirected Graphical Models

Whiteboard

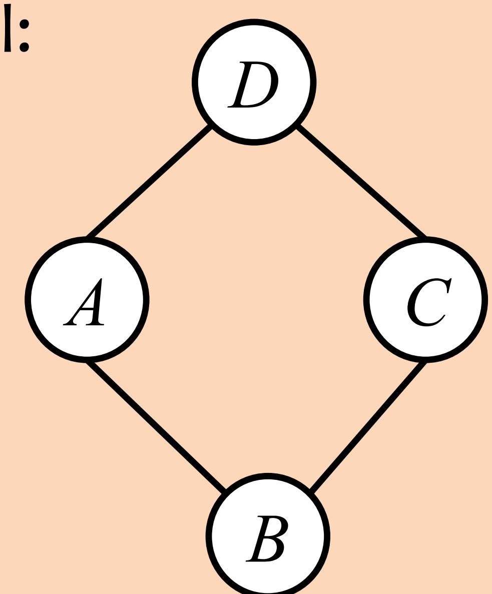
– Proof of independence by separation (simple case)

Non-equivalence of Directed / Undirected Graphical Models

There does **not** exist an **undirected** graphical model that can capture the conditional independence assumptions of this **directed** graphical model:



There does **not** exist a **directed** graphical model that can capture the conditional independence assumptions of this **undirected** graphical model:



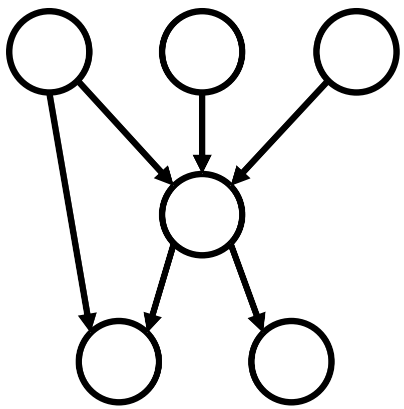
Exercise: Can you prove these claims?

Representation of both directed and undirected graphical models

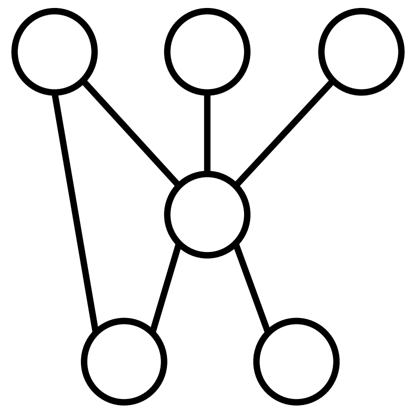
FACTOR GRAPHS

Three Types of Graphical Models

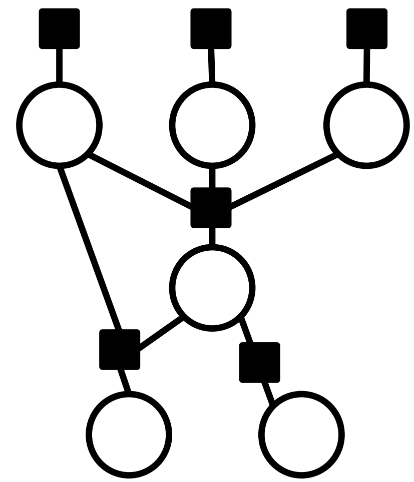
Directed Graphical Model



Undirected Graphical Model



Factor Graph



How General Are Factor Graphs?

- Factor graphs can be used to describe
  - Markov Random Fields (undirected graphical models)
  - Conditional Random Fields
  - Bayesian Networks (directed graphical models)

Factor Graph Notation

- Variables:

$$\mathcal{X} = \{X_1, \ldots, X_i, \ldots, X_n\}$$

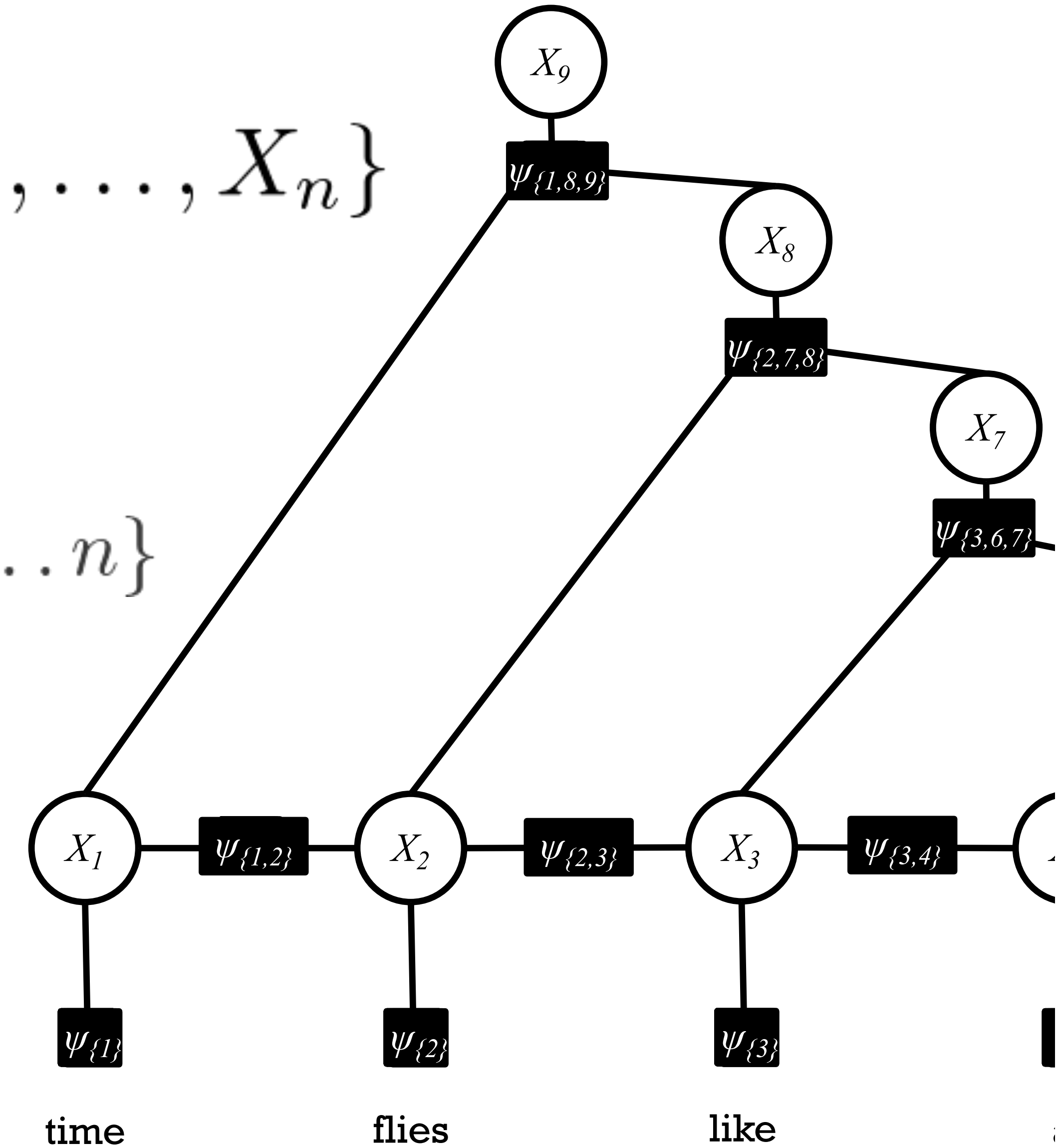
- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

where $\alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\}$

Joint Distribution

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_\alpha \psi_\alpha(\boldsymbol{x}_\alpha)$$



Factors are Tensors

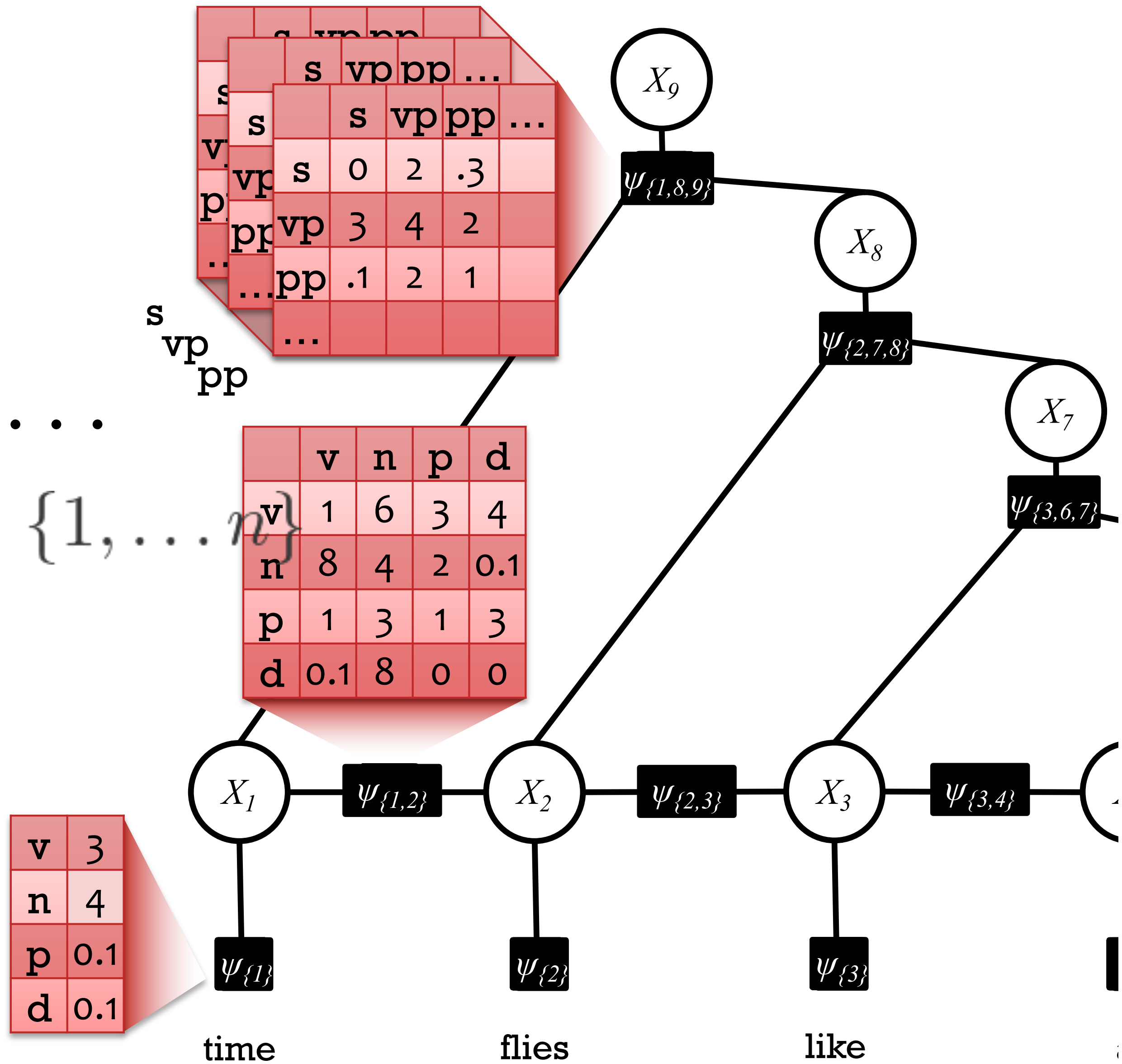
- *Def*: the **arity** of a factor is the number of neighbors (variables) it has

- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

where $\alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\}$

- *Def*: a **unary factor** touches one variables
- *Def*: a **binary factor** touches two variables
- *Def*: a **ternary factor** touches three variables



	s	vp	pp	…
s	0	2	.3	
vp	3	4	2	
pp	.1	2	1	
…				

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

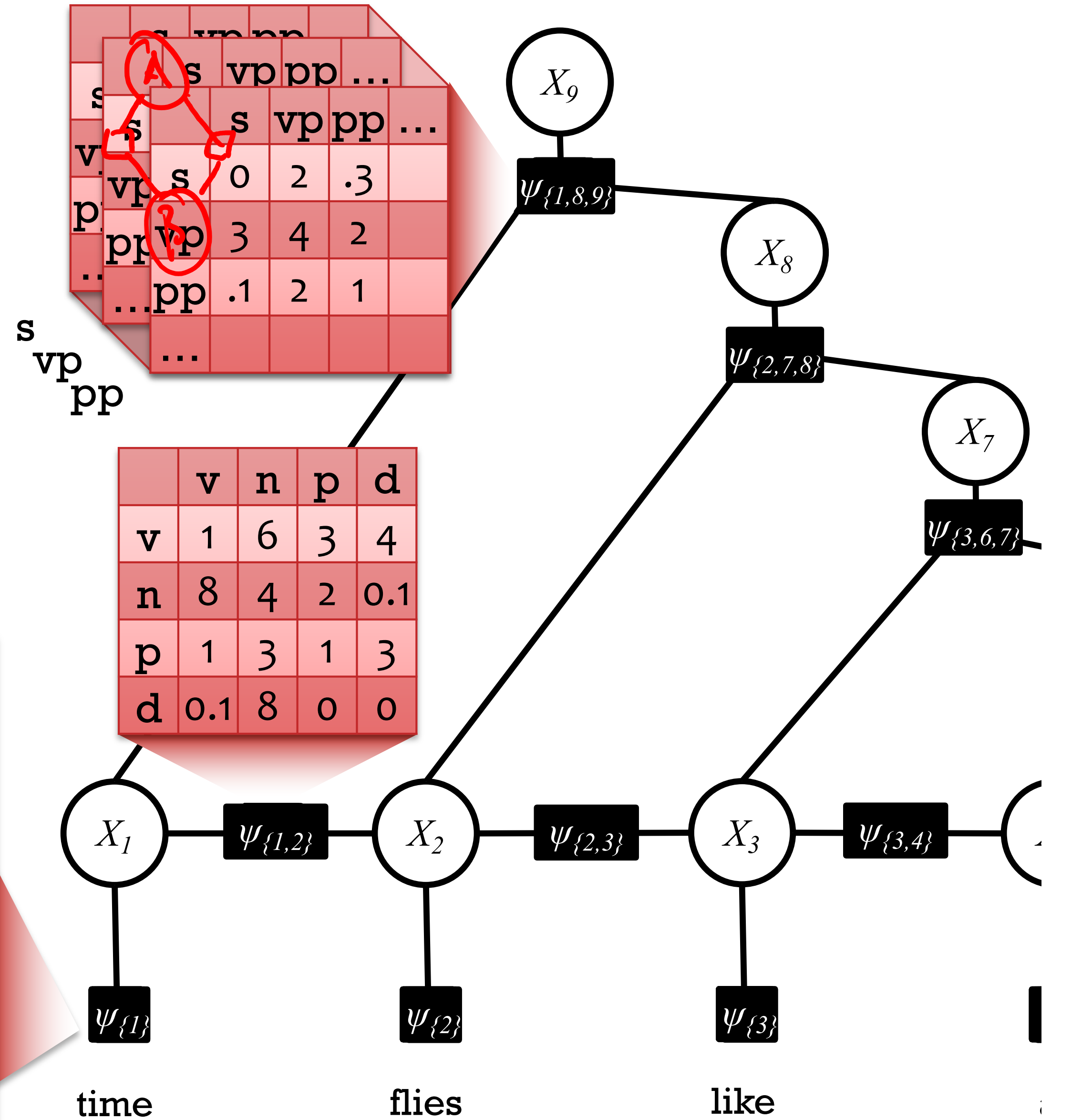
v	3
n	4
p	0.1
d	0.1

Factors are Tensors

- Factors must contain **non-negative** values -- this ensures we have a valid probability distribution
- We also sometimes refer to factors as **potential functions** or **potentials** (like UGMs)

Joint Distribution

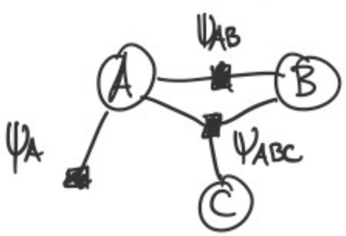
$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$



	s	vp	pp	…
s	0	2	.3	
vp	3	4	2	
pp	.1	2	1	
…				

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

Ex: Factor Graph over Binary Variables



a	$\psi_A(a)$
0	2
1	7

a	b	$\psi_{AB}(a,b)$
0	0	4
0	1	3
1	0	6
1	1	2

a	b	c	$\psi_{ABC}(a,b,c)$
0	0	0	6
0	0	1	2
0	1	0	1
	...		...
1	1	1	5

$$P(A=a, B=b, C=c) = p(a,b,c) = \frac{1}{Z} \underbrace{\psi_A(a)\, \psi_{AB}(a,b)\, \psi_{ABC}(a,b,c)}_{s(a,b,c)} \Rightarrow Z = \sum_a \sum_b \sum_c s(a,b,c)$$

a	b	c	ψ_A	ψ_{AB}	ψ_{ABC}	$s(\cdot)$	$p(\cdot)$
0	0	0	2	4	6	48	48/Z
0	0	1	2	4	2	16	16/Z
0	1	0	2	3	1	6	6/Z
	...			...		⋮	
1	1	1	7	2	5	+ 70	70/Z
						Z	

CONVERTING UGMS AND DGMS TO FACTOR GRAPHS

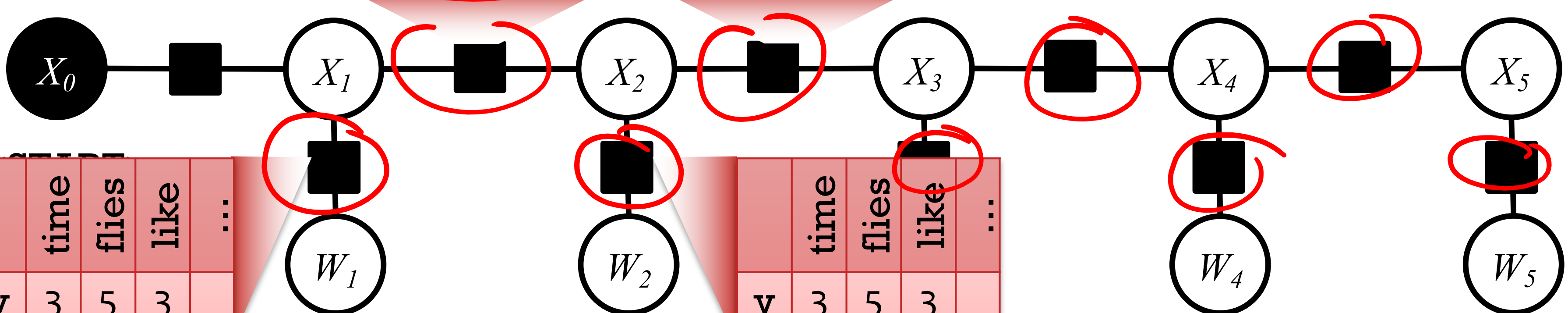
Factors have local opinions (≥ 0)

Each black box looks at *some* of the tags X_i and words W_i

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

Note: We chose to reuse the same factors at different positions in the sentence.



	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

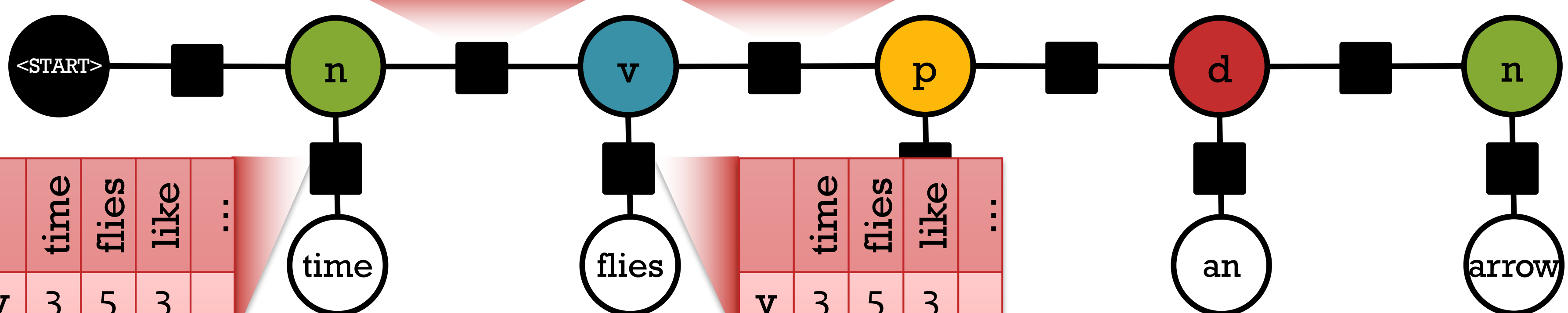
Factors have local opinions (≥ 0)

Each black box looks at *some* of the tags X_i and words W_i

$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) \quad = \quad ?$$

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0



	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

Global probability = product of local opinions

Each black box looks at *some* of the tags X_i and words W_i

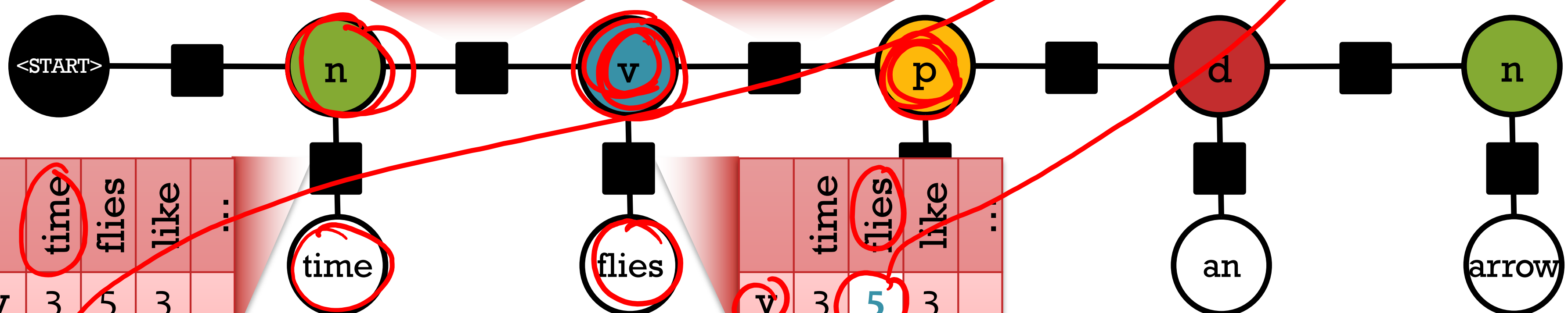
$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = \frac{1}{Z}(4 * 8 * 5 * 3 * \ldots)$$

Uh-oh! The probabilities of the various assignments sum up to Z > 1.

So divide them all by Z.

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0



	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

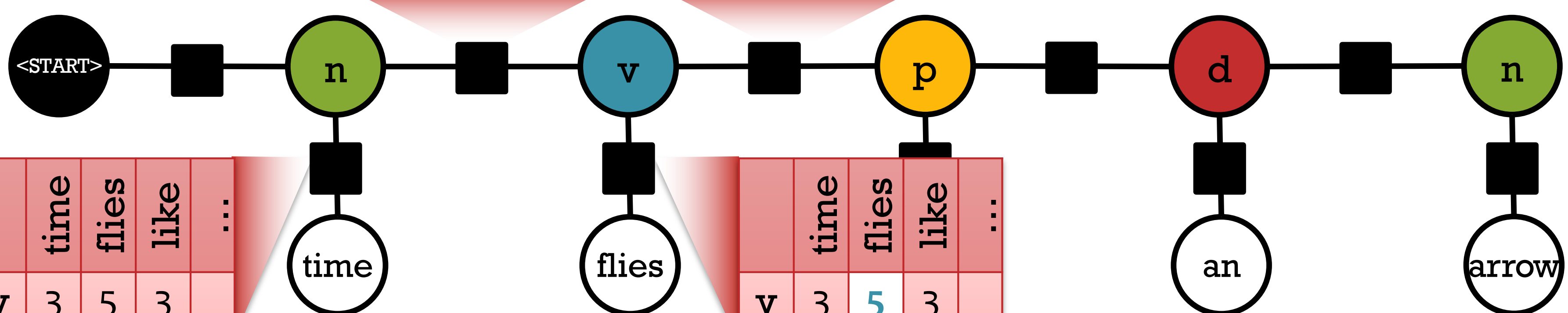
Markov Random Field (MRF)

Joint distribution over tags X_i <u>and</u> words W_i
The individual factors aren't *necessarily* probabilities.

$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = \frac{1}{Z}(4 * 8 * 5 * 3 * \ldots)$$

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0



	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

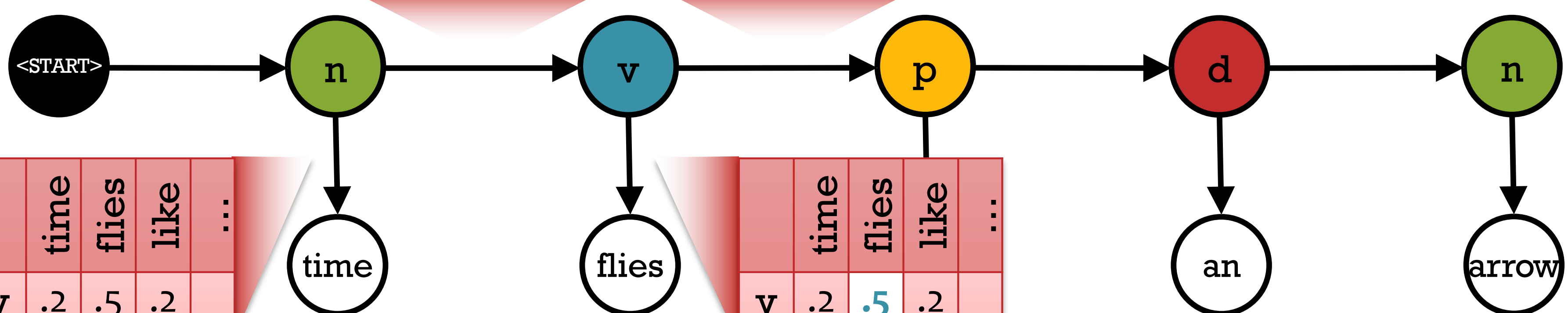
Bayesian Networks

But sometimes we *choose* to make them probabilities.
Constrain each row of a factor to sum to one. Now $Z = 1$.

$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = \cancel{\frac{1}{Z}}(.3 * .8 * .2 * .5 * \ldots)$$

	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

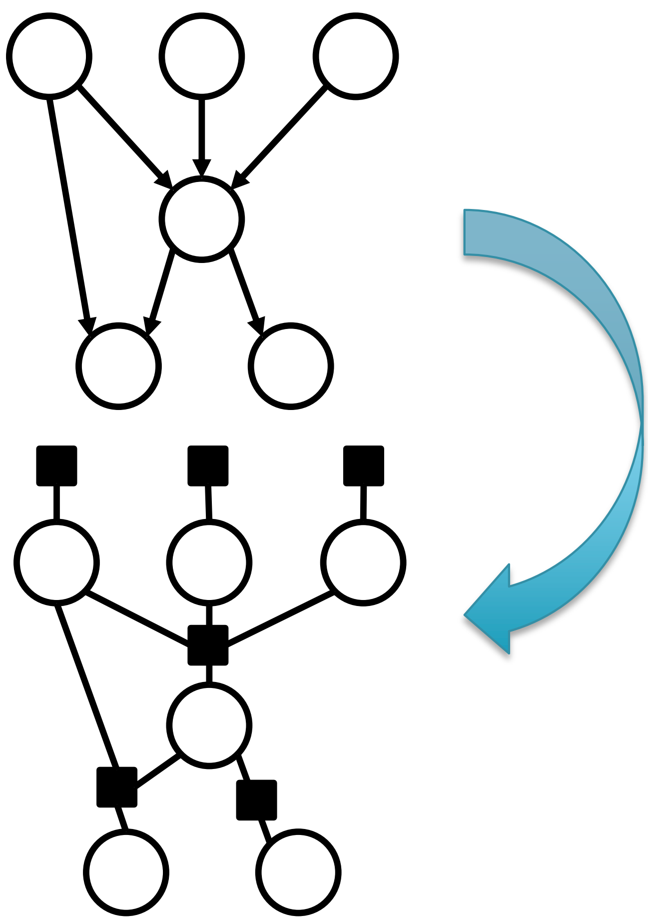


	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

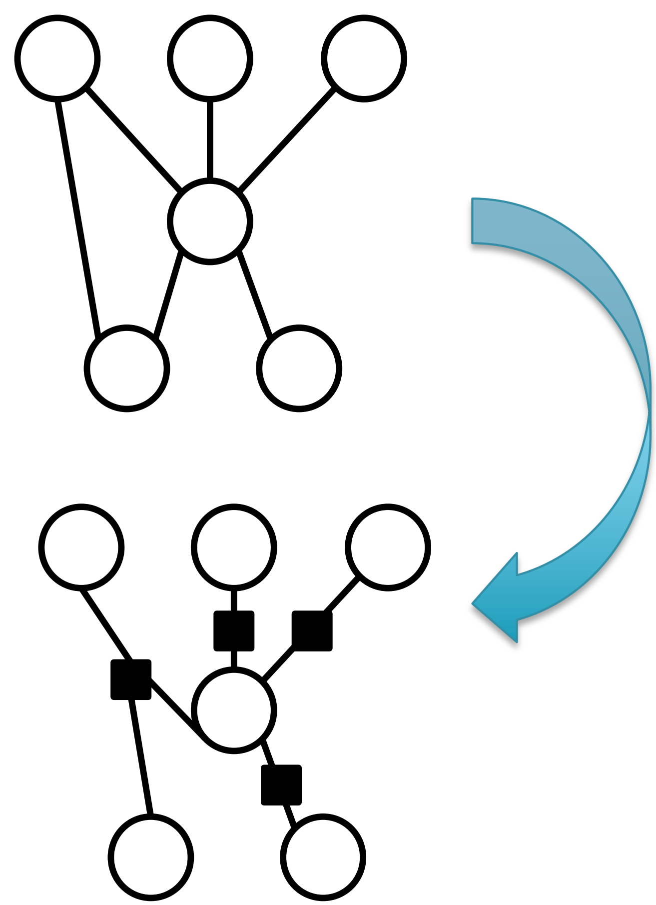
	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

Converting to Factor Graphs

Each conditional and marginal distribution in a **directed GM** becomes a factor

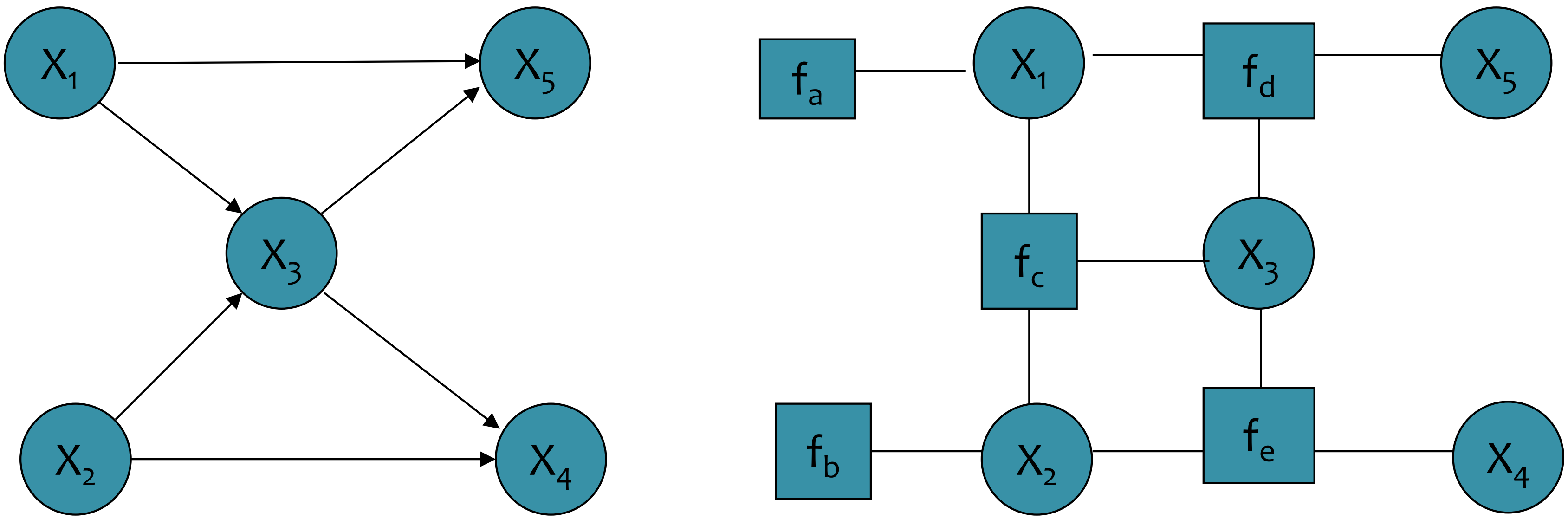


Each maximal clique (or each clique) in an **undirected GM** becomes a factor



Converting to Factor Graphs

- Example 1

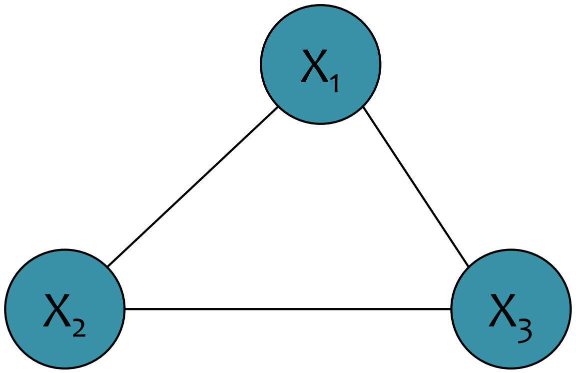


$P(X_1)$ $P(X_2)$ $P(X_3|X_1,X_2)$ $P(X_5|X_1,X_3)$ $P(X_4|X_2,X_3)$

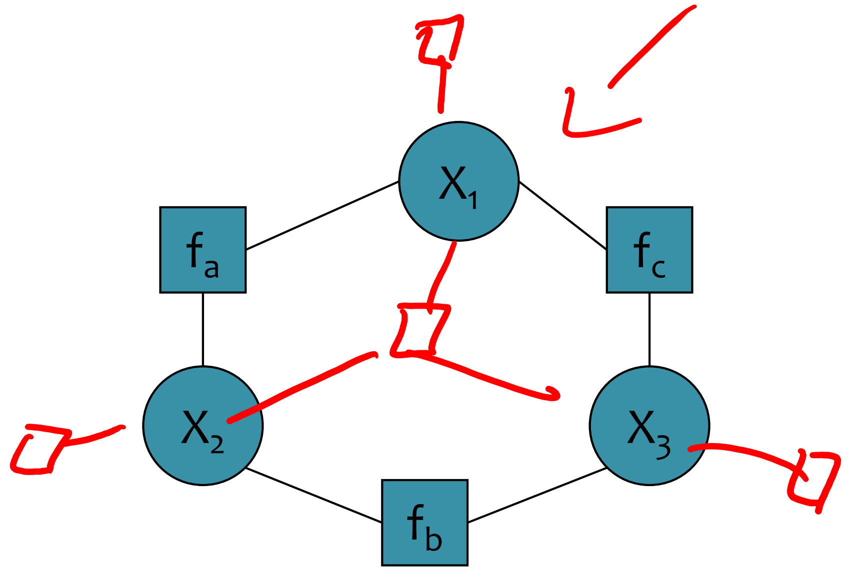
$f_a(X_1)$ $f_b(X_2)$ $f_c(X_3,X_1,X_2)$ $f_d(X_5,X_1,X_3)$ $f_e(X_4,X_2,X_3)$

Converting to Factor Graphs

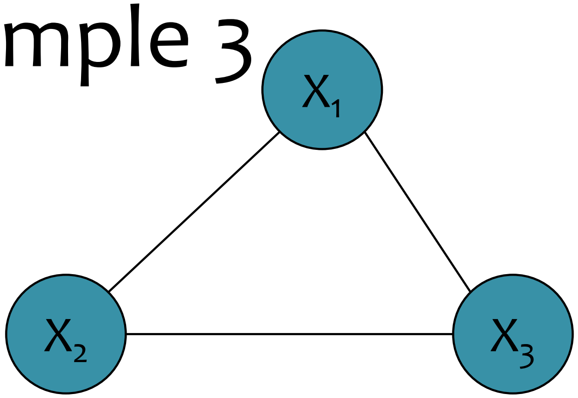
- Example 2



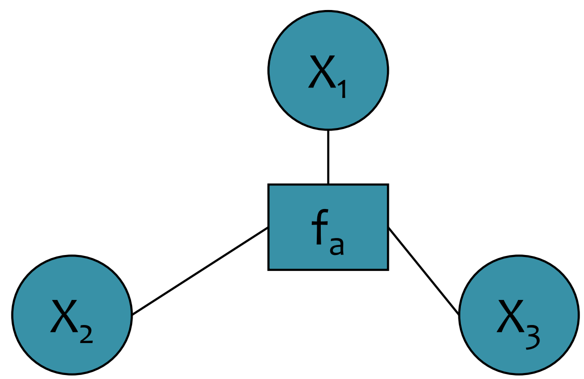
$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2)f_b(x_2,x_3)f_c(x_3,x_1)$$



- Example 3

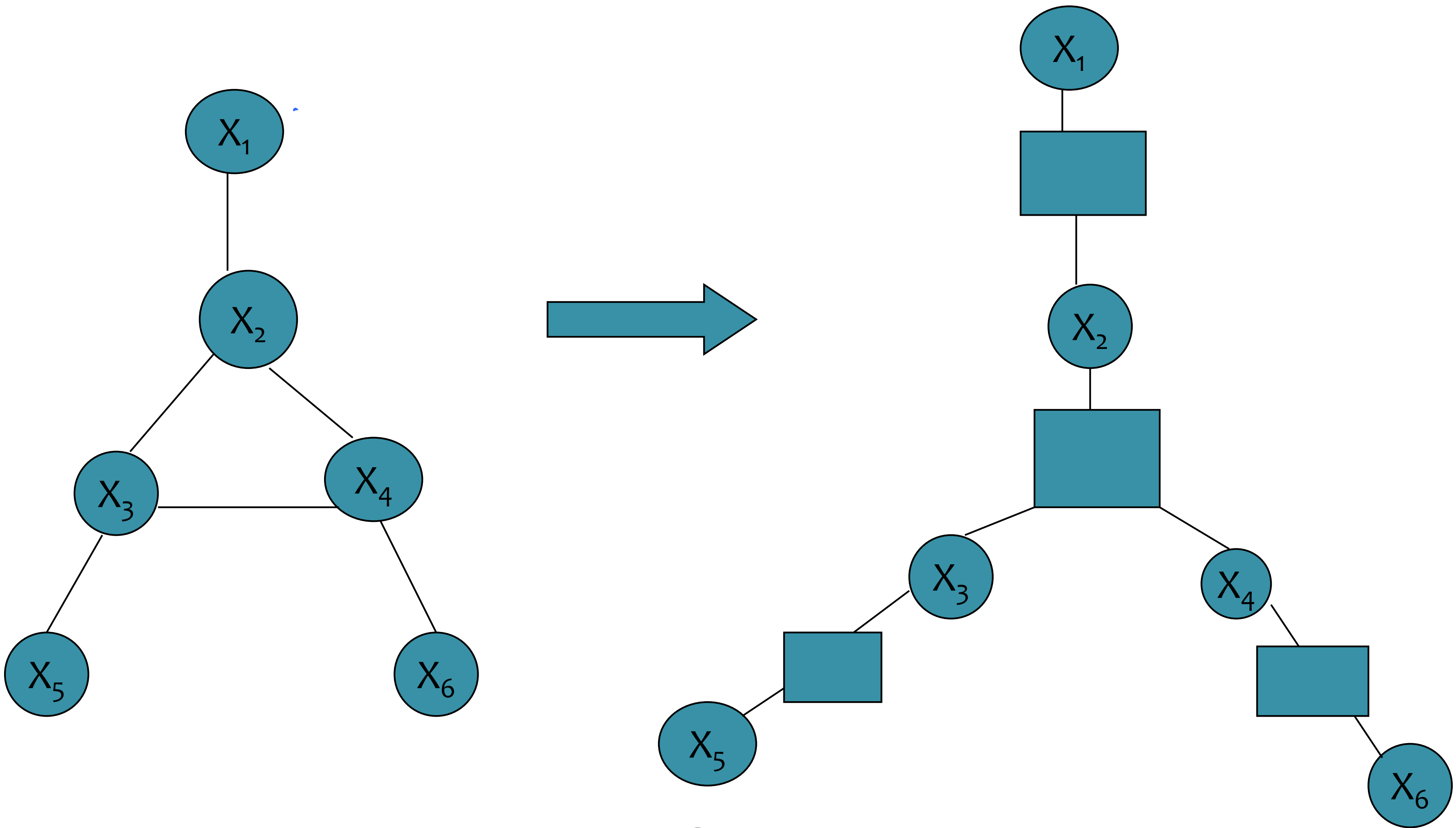


$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2,x_3)$$



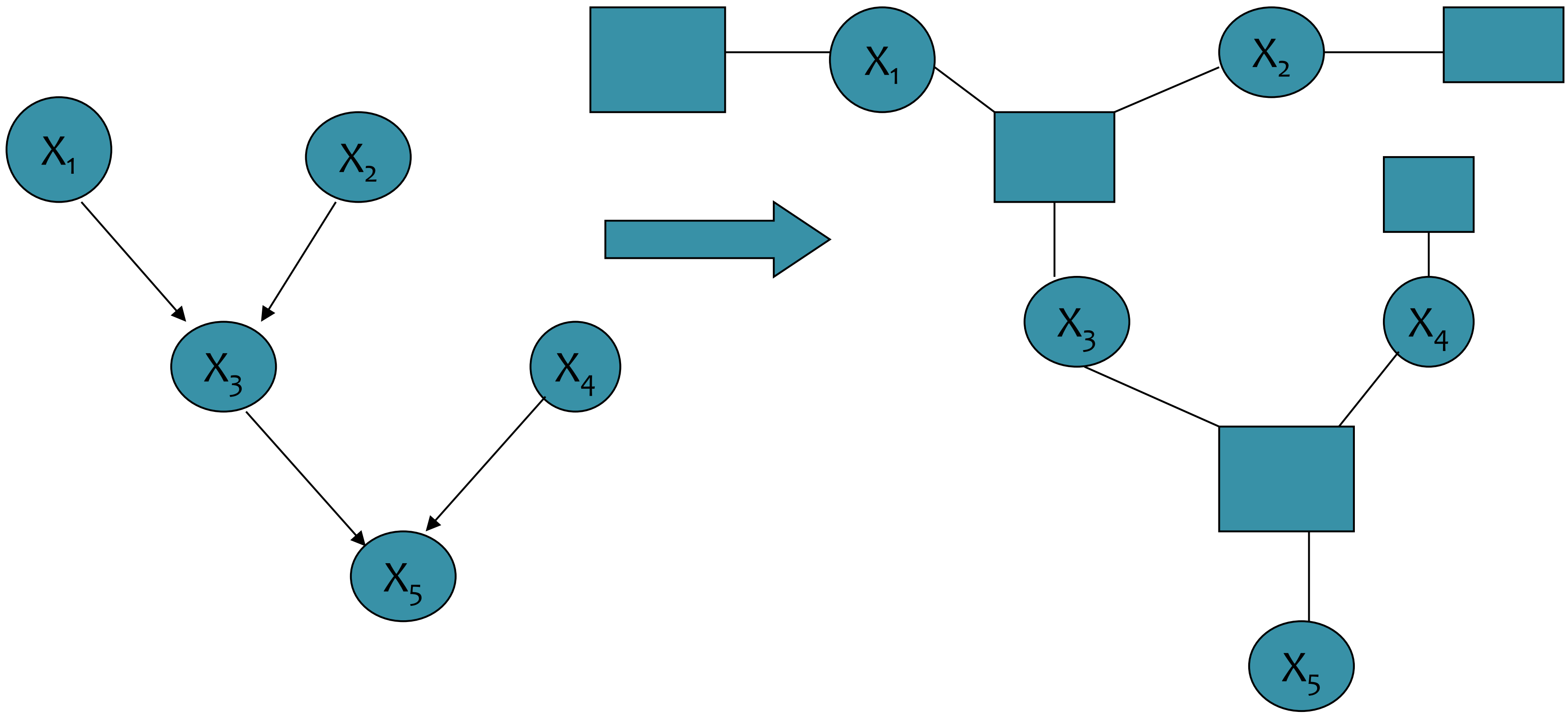
Converting to Factor Graphs

- Example 4

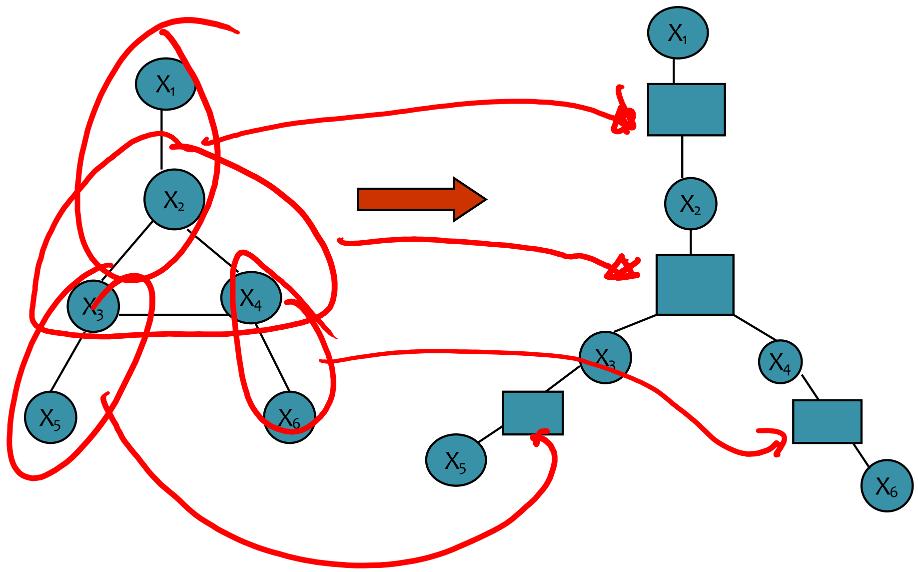


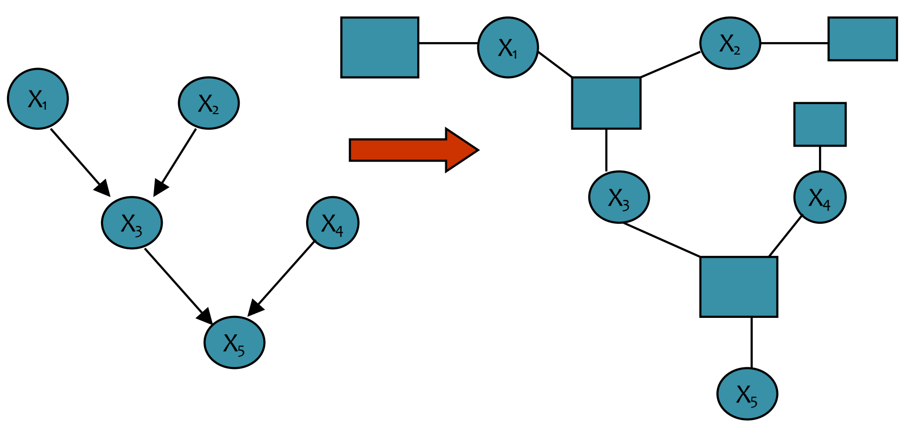
Converting to Factor Graphs

- Example 5



Converting to Factor Graphs





A neat property of factor graph convertion:

- A factor graph sometimes turns tree-like undirected / directed graphical models to factor trees,
- Trees are a data-structure that guarantees exactness of belief propagation!

Converting to Factor Graphs

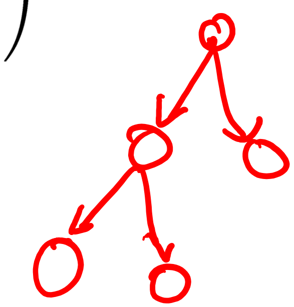
Equivalence of directed and undirected trees

- Any undirected tree can be converted to a directed tree by choosing a root node and directing all edges away from it
- A directed tree and the corresponding undirected tree make the same conditional independence assertions
- Parameterizations are essentially the same.
  - Undirected tree:
  - Directed tree:
  - Equivalence:

$$p(x) = \frac{1}{Z}\left(\prod_{i\in V}\psi(x_i)\prod_{(i,j)\in E}\psi(x_i,x_j)\right)$$

$$p(x) = p(x_r)\prod_{(i,j)\in E}p(x_j|x_i)$$

$$\psi(x_r) = p(x_r);\quad \psi(x_i,x_j) = p(x_j|x_i);$$
$$Z = 1,\quad \psi(x_i) = 1$$



MRF VS. CRF

MRF vs. CRF

Markov Random Field (MRF):

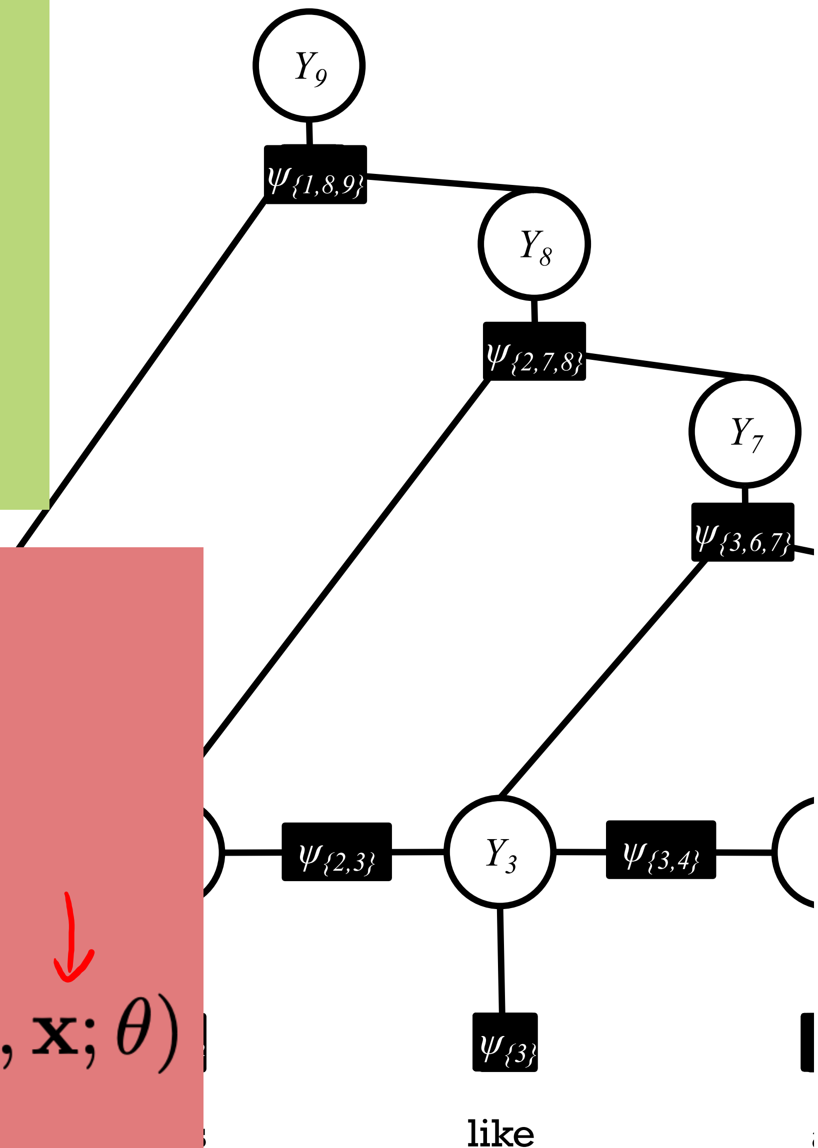
- just a distribution over variables **y**
- partition function Z is just a function of the parameters

$$p_{\boldsymbol{\theta}}(\mathbf{y}) = \frac{1}{Z(\theta)} \prod_{\alpha} \psi_{\alpha}(\mathbf{y}_{\alpha}; \theta)$$

Conditional Random Field (CRF):

- conditions on some additional observed variables **x**
- partition function Z is a function of **x** as well

$$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x}; \theta)} \prod_{\alpha} \psi_{\alpha}(\mathbf{y}_{\alpha}, \mathbf{x}; \theta)$$



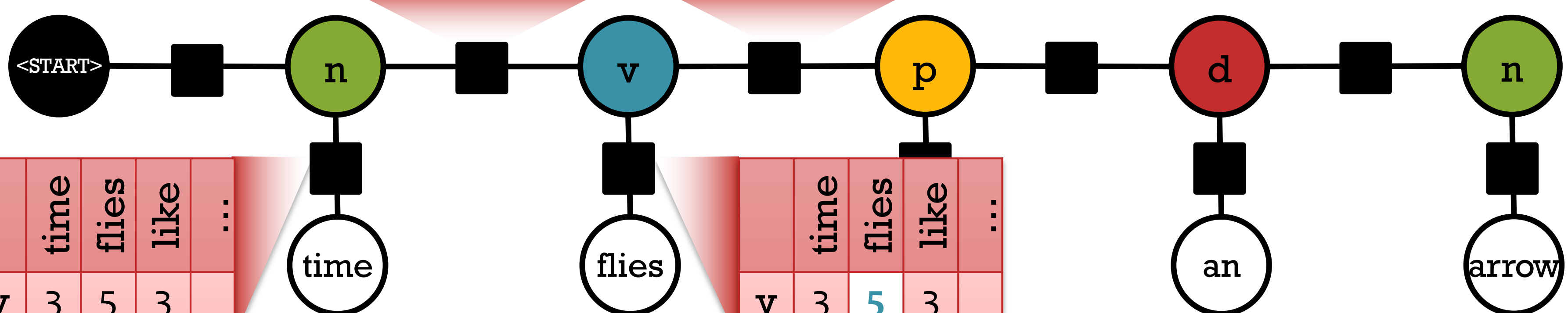
Markov Random Field (MRF)

Joint distribution over tags X_i and words W_i

$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = \frac{1}{Z}(4 * 8 * 5 * 3 * \ldots)$$

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0



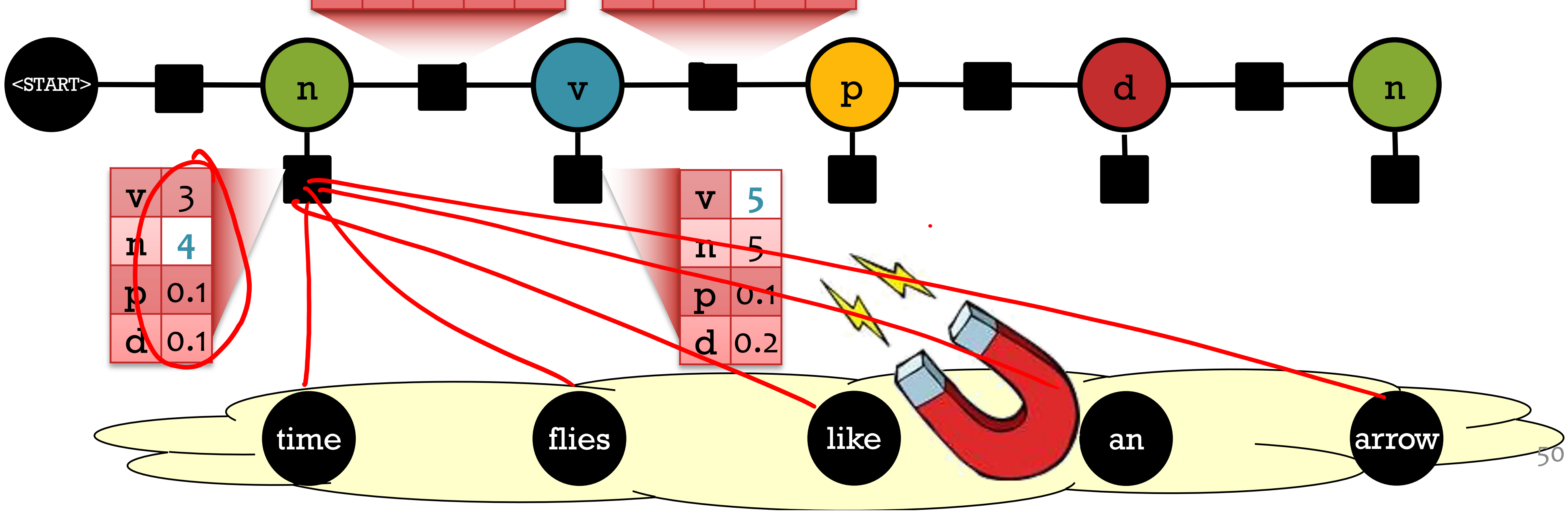
	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

Conditional Random Field (CRF)

Conditional distribution over tags X_i given words w_i.
The factors and Z are now specific to the sentence w.

$$p(\text{n, v, p, d, n} \mid \text{time, flies, like, an, arrow}) = \frac{1}{Z}(4 * 8 * 5 * 3 * \ldots)$$



	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

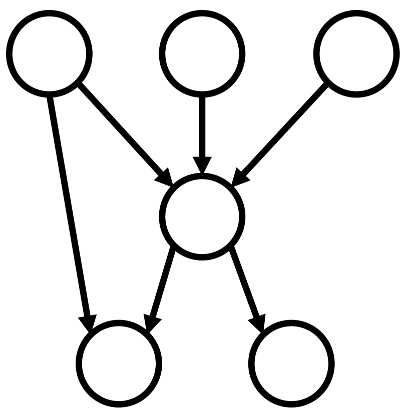
v	3
n	4
p	0.1
d	0.1

v	5
n	5
p	0.1
d	0.2

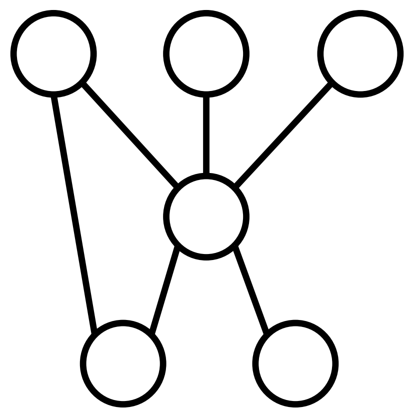
TYPES OF GRAPHICAL MODELS

Three Types of Graphical Models

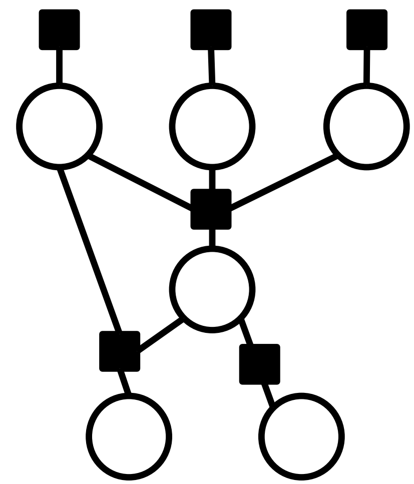
Directed Graphical Model



Undirected Graphical Model



Factor Graph



Key Concepts for Graphical Models

Graphical Models in General

1. A graphical model defines a **family of probability distributions**
2. That family shares in common a set of **conditional independence assumptions**
3. By choosing a **parameterization** of the graphical model, we obtain a single **model** from the family
4. The model may be either **locally or globally normalized**

Ex: Directed G.M.

1. **Family**: directed graphs with locally normalized conditional probabilities
2. **Conditional Independencies**: d-separation, Markov blanket
3. **Example parameterization**: conditional probability tables (CPTs) for discrete var.s, conditional probability densities for continuous var.s
4. **Normalization**: locally normalized, partition function is always 1.0

Key Concepts for Graphical Models

Graphical Models in General

1. A graphical model defines a **family of probability distributions**
2. That family shares in common a set of **conditional independence assumptions**
3. By choosing a **parameterization** of the graphical model, we obtain a single **model** from the family
4. The model may be either **locally or globally normalized**

Ex: Undirected G.M.

1. **Family:** undirected graphs with unormalized potentials
2. **Conditional Independencies:** independence by separation, Markov blanket
3. **Example parameterization:** Markov random field (MRF), conditional random field (CRF), neural potentials
4. **Normalization:** globally normalized

Key Concepts for Graphical Models

Graphical Models in General

1. A graphical model defines a **family of probability distributions**
2. That family shares in common a set of **conditional independence assumptions**
3. By choosing a **parameterization** of the graphical model, we obtain a single **model** from the family
4. The model may be either **locally or globally normalized**

Ex: Factor Graph

1. **Family:** bipartite graph over variables and factors
2. **Conditional Independencies:** independence by separation, inferable from underlying DGM or UGM
3. **Example parameterization:** any DGM parameterization, any UGM parameterization
4. **Normalization:** locally normalized if based on DGM, globally normalized if based on UGM

Q1: what Qs?

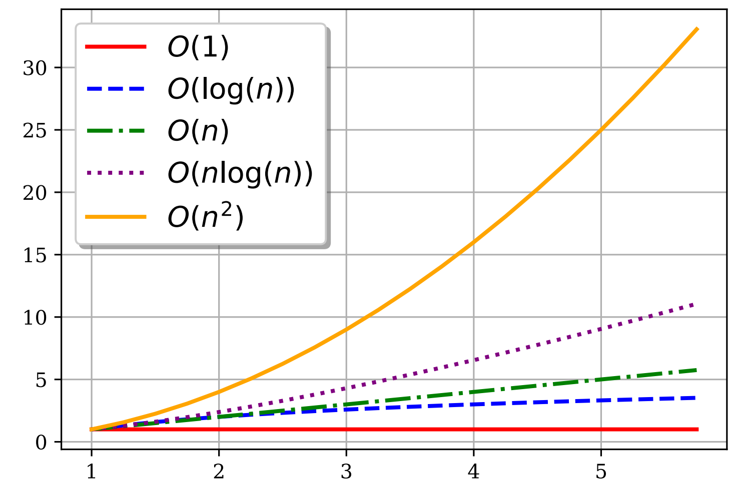
Q&A

COMPUTATIONAL COMPLEXITY

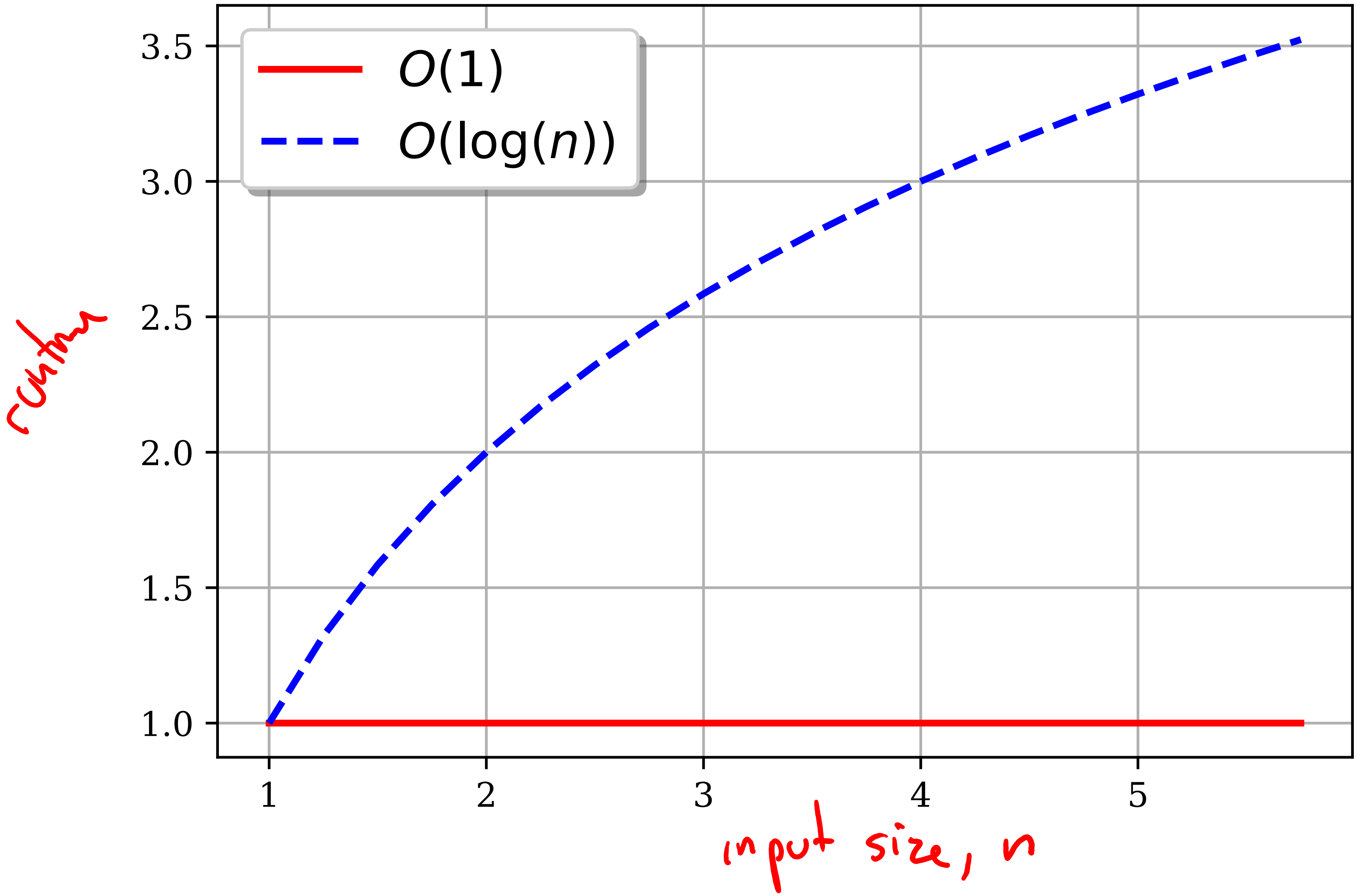
Analysis of Algorithms

Key Questions:

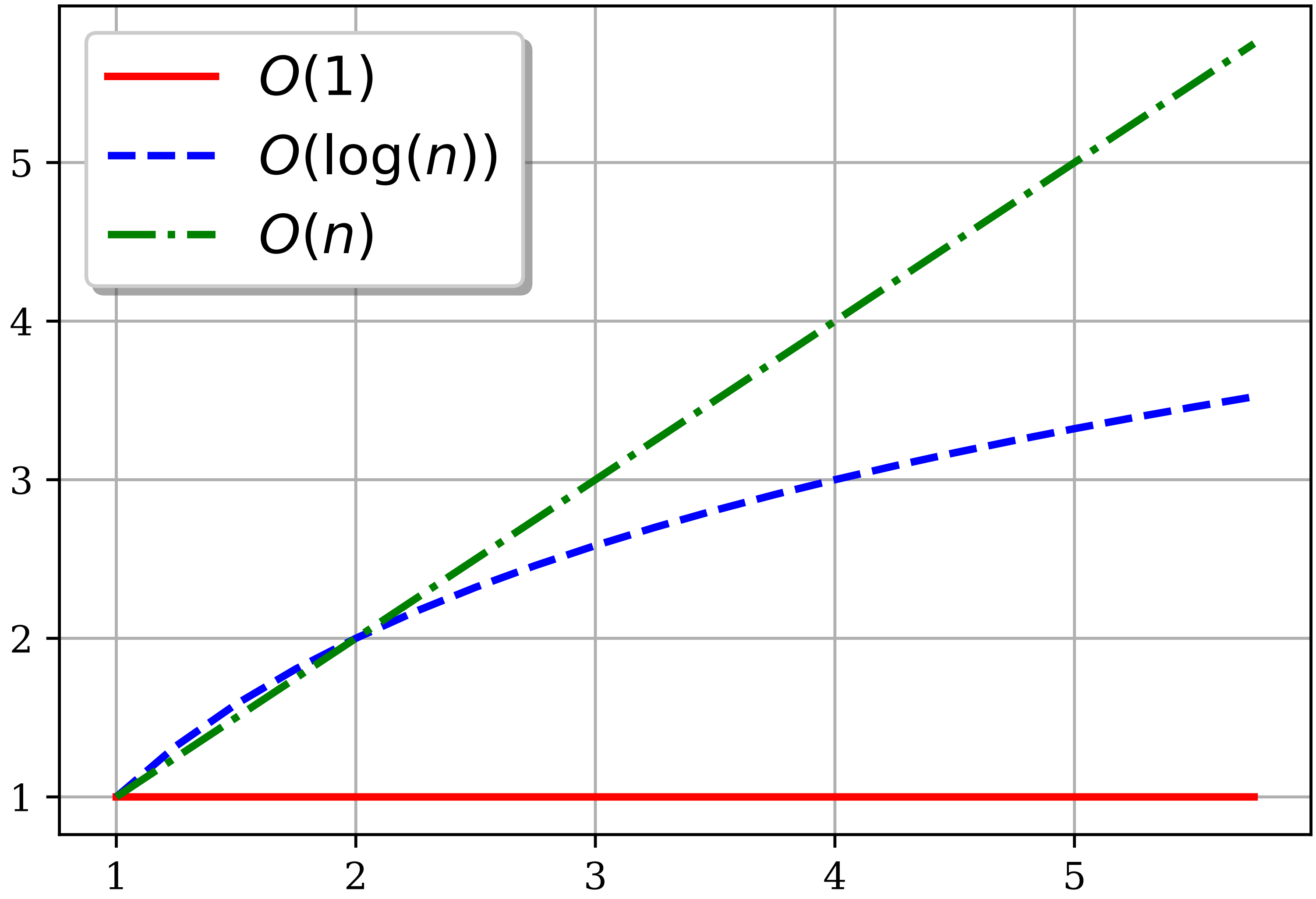
1. Given a single algorithm, will it complete on a given input in a reasonable amount of time/space?
2. Given two algorithms which one is better?



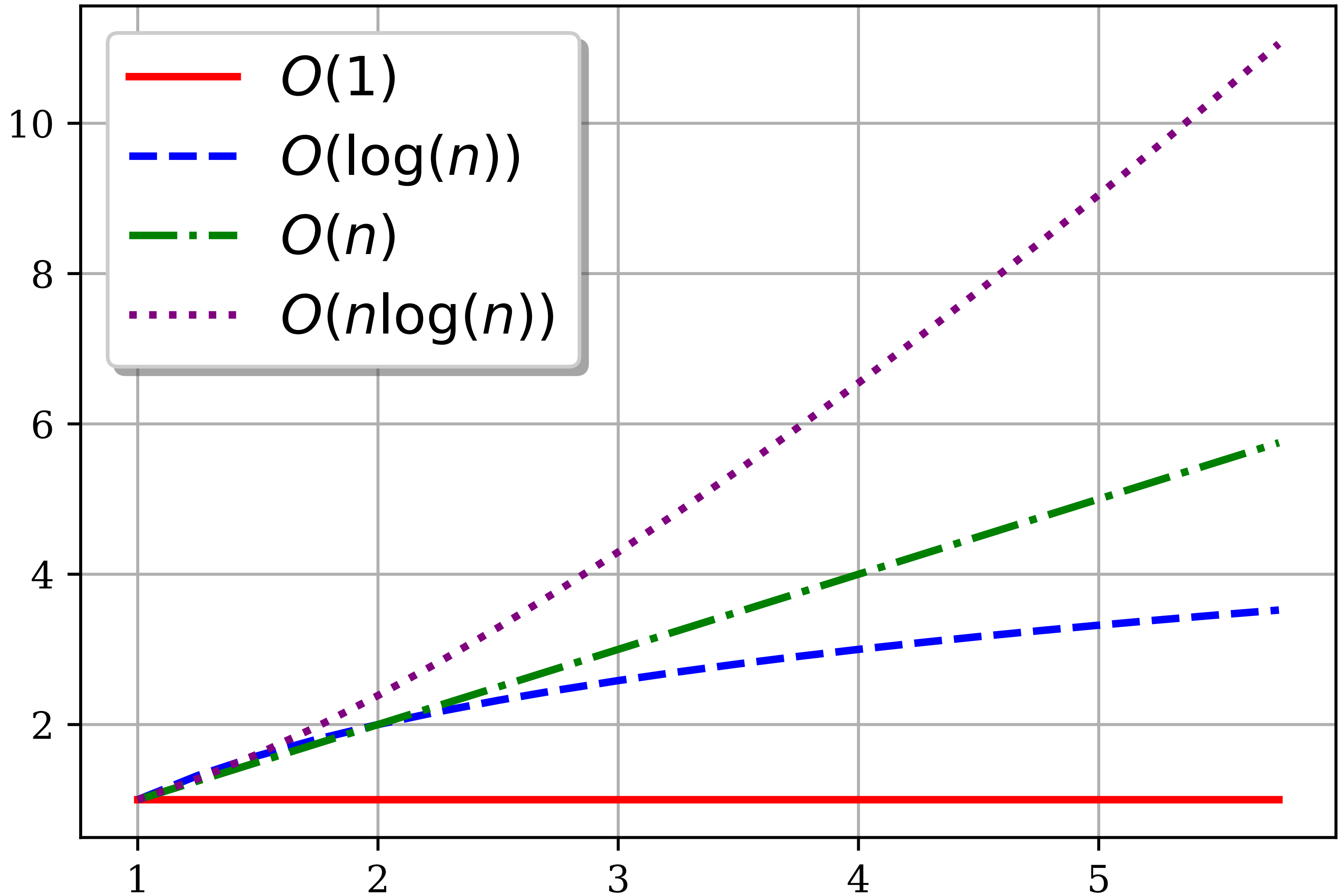
Comparing Algorithm Runtimes



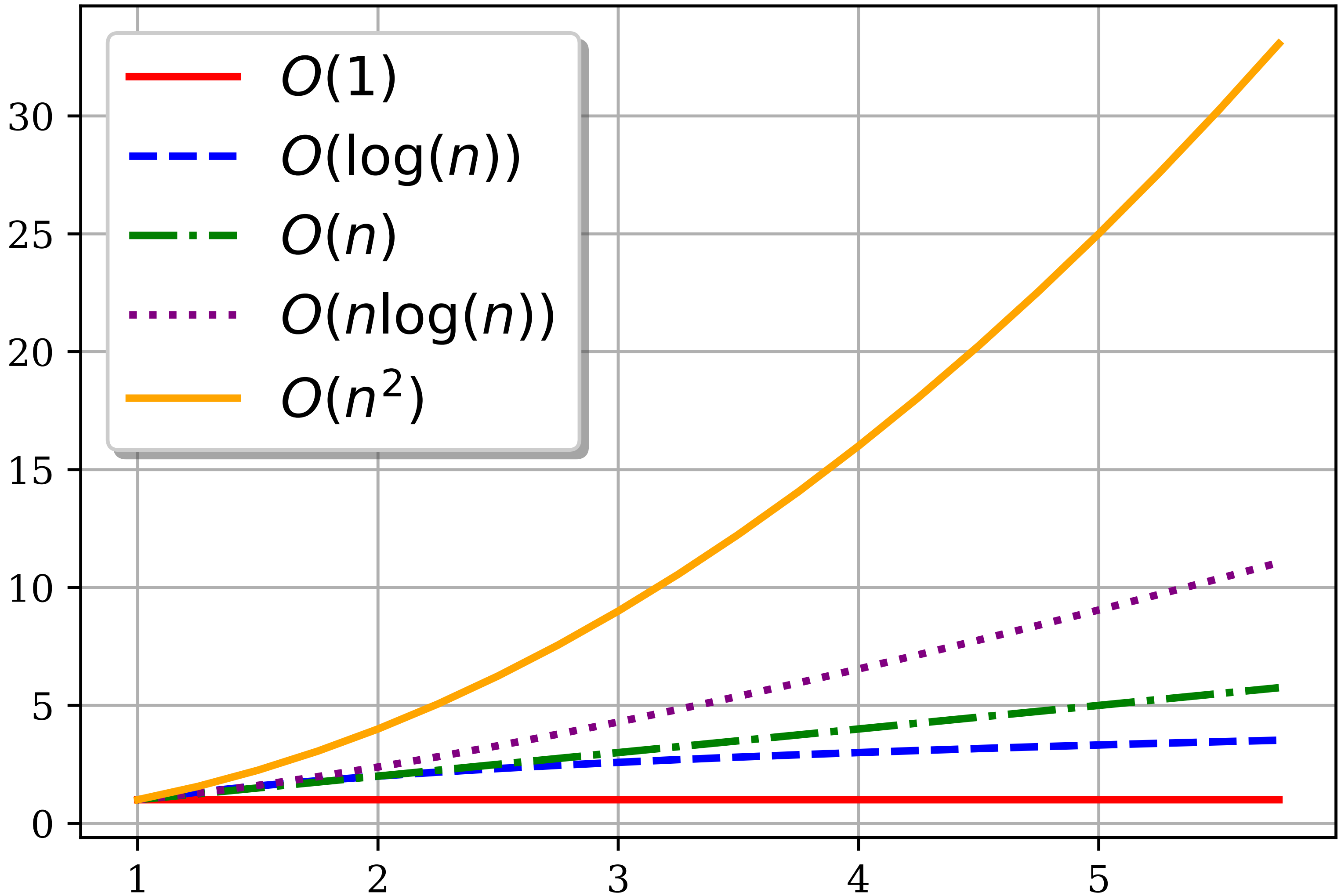
Comparing Algorithm Runtimes



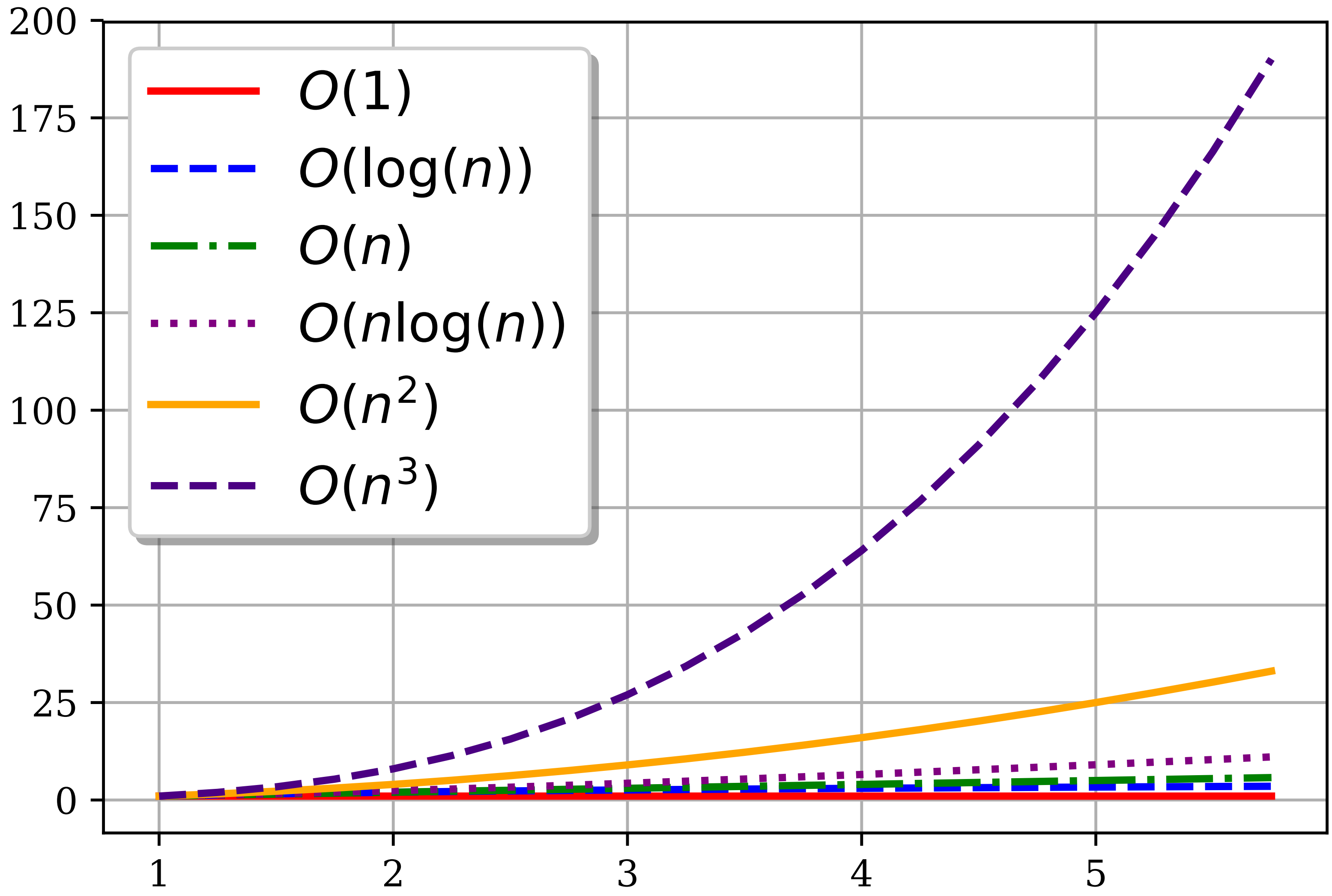
Comparing Algorithm Runtimes



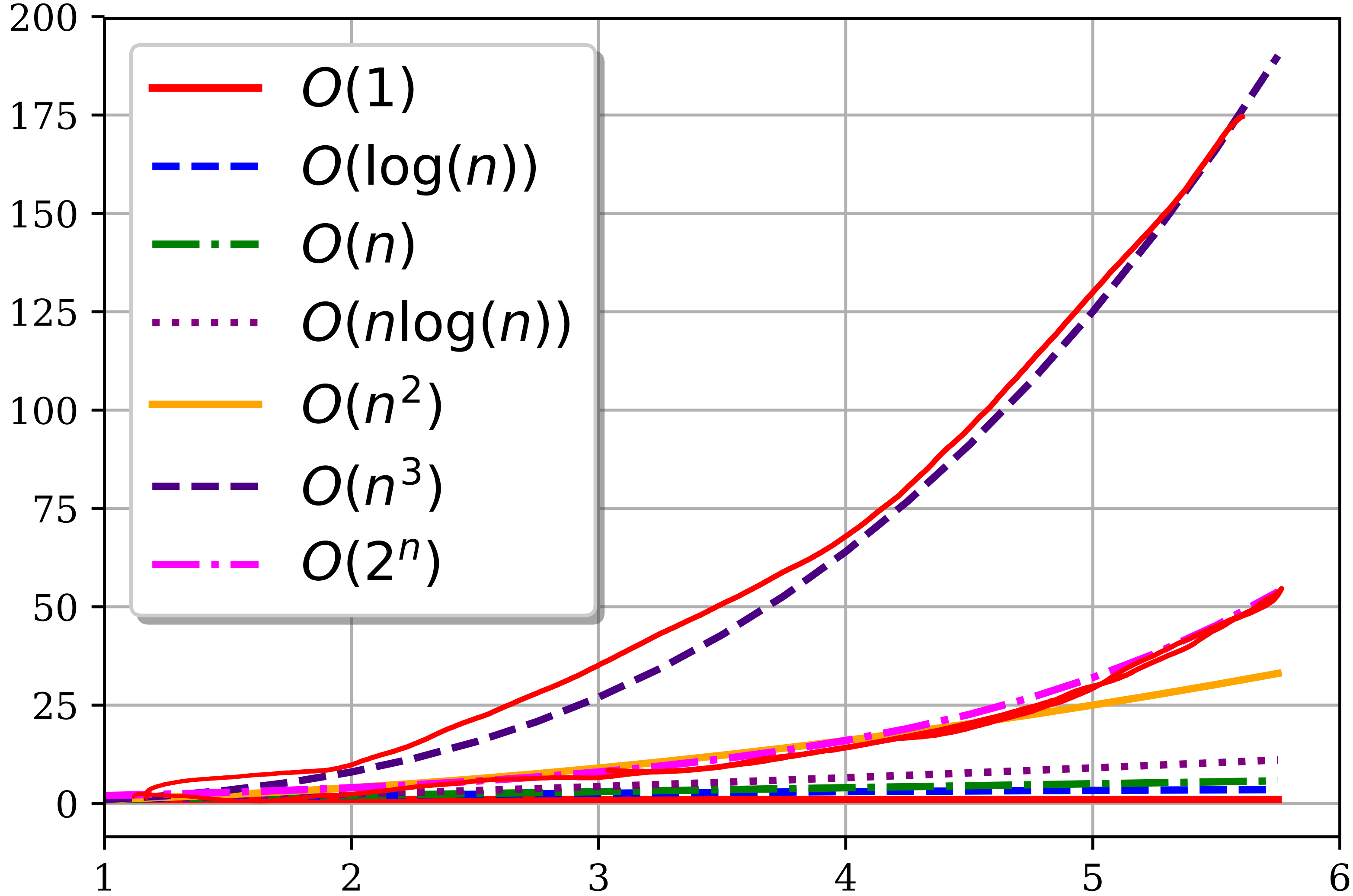
Comparing Algorithm Runtimes



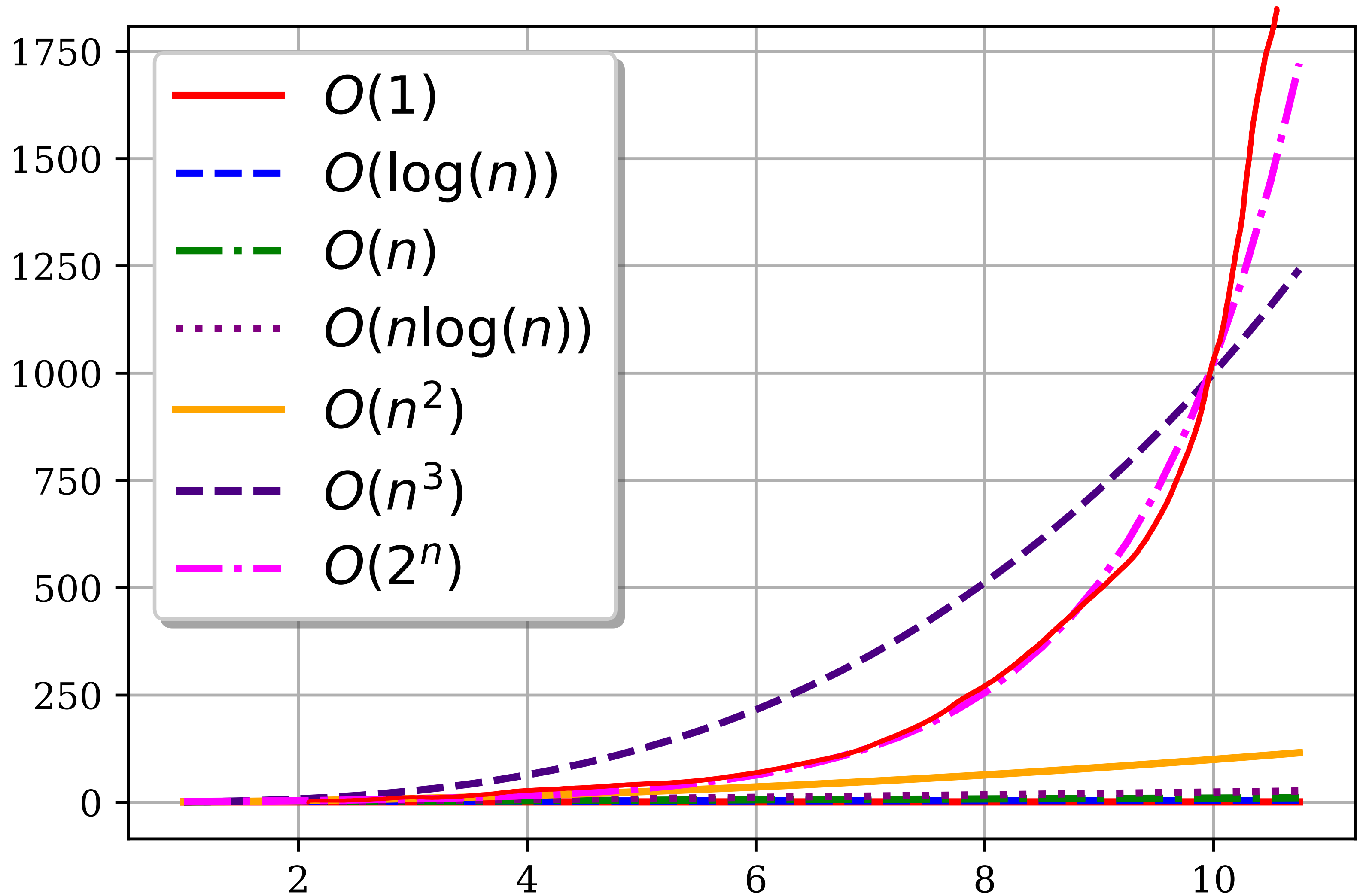
Comparing Algorithm Runtimes



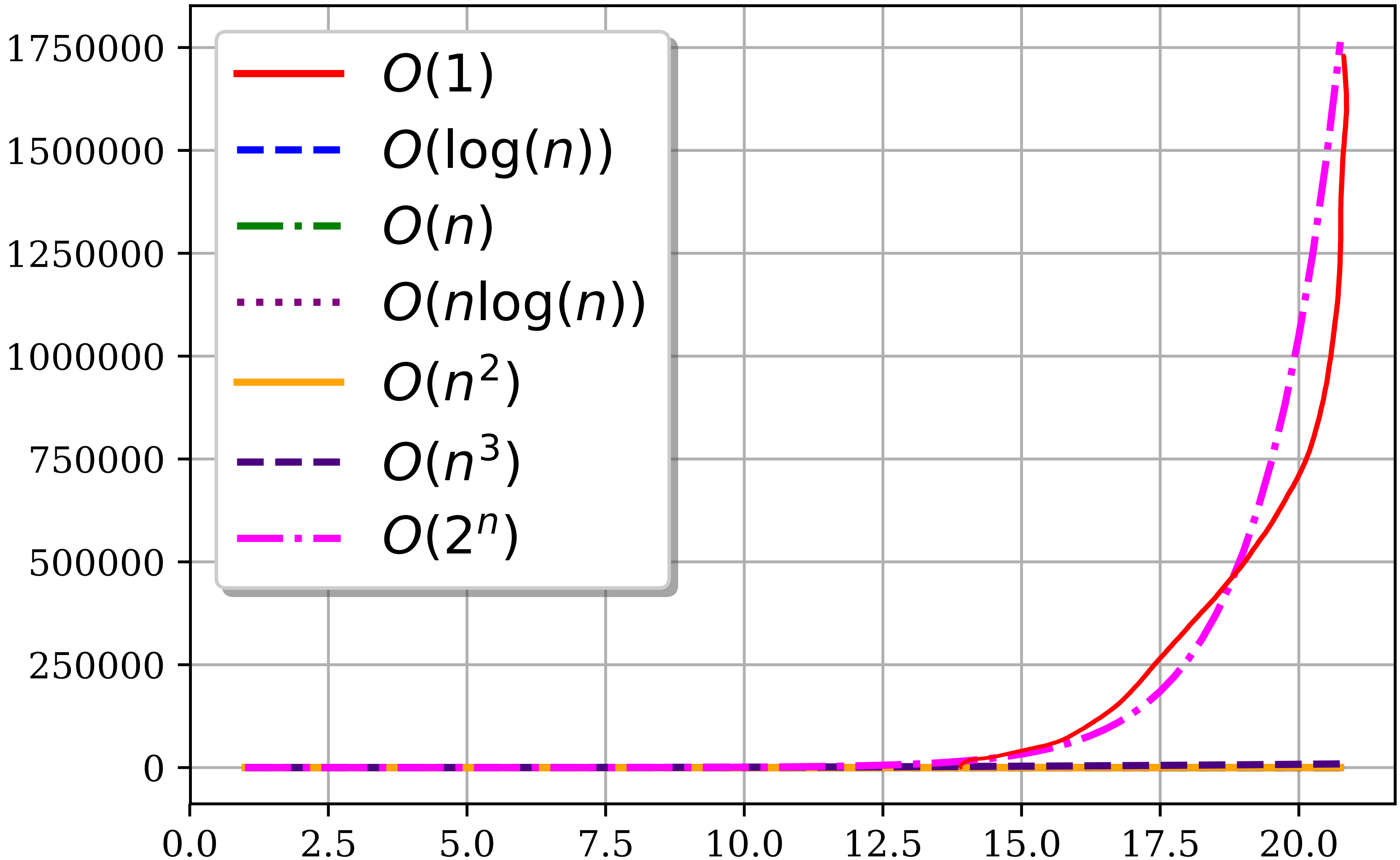
Comparing Algorithm Runtimes



Comparing Algorithm Runtimes



Comparing Algorithm Runtimes



Comparing Algorithm Runtimes

Computational Complexity	Name
$O(1)$	constant
$O(\log(n))$	logarithmic
$O(n)$	linear
$O(n \log(n))$	"n log n"
$O(n^2)$	quadratic
$O(n^3)$	cubic
$O(2^n)$	exponential
$O(n!)$	factorial
$O(n^n)$	superexponential

Complexity Classes

- An algorithm runs in **polynomial time** if its runtime is a polynomial function of the input size (e.g. $O(n^k)$ for some fixed constant k)
- The **class P** consists of all problems that can be solved in polynomial time

- A problem for which the answer is binary (e.g. yes/no) is called a **decision problem**
- The **class NP** contains all decision problems where 'yes' answers can be verified (proved) in polynomial time
- A problem is **NP-Hard** if given an O(1) oracle to solve it, every problem in NP can be solved in polynomial time (e.g. by reduction)
- A problem is **NP-Complete** if it belongs to both the classes NP and NP-Hard

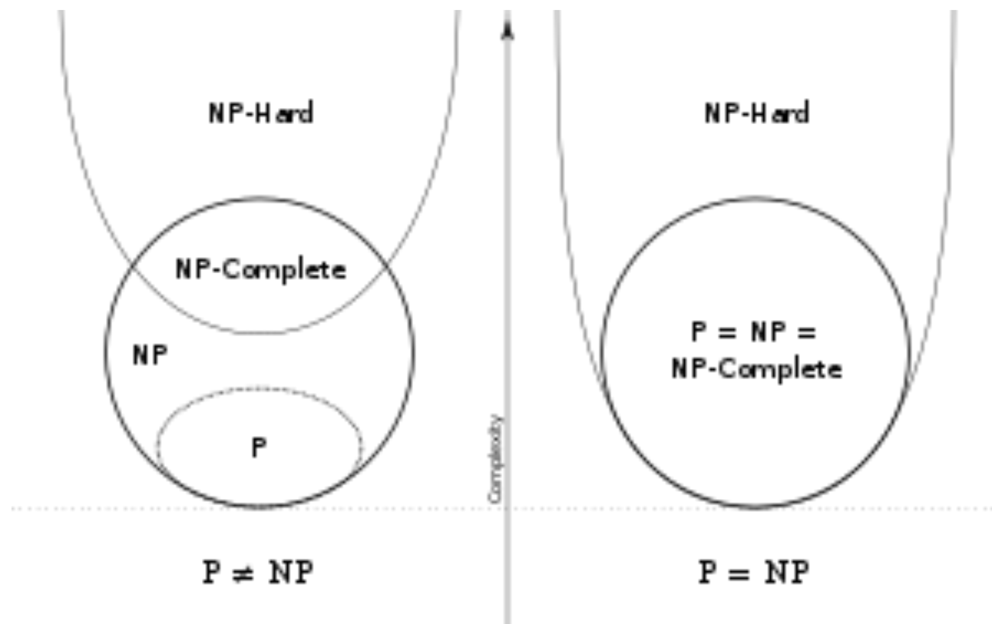


Figure from https://en.wikipedia.org/wiki/NP-completeness

Complexity Classes

- A problem for which the answer is a nonnegative integer is called a **counting problem**
- The **class #P** contains the counting problems that align to decision problems in NP
  - really this is the class of problems that count the number of accepting paths in a Turing machine that is nondeterministic and runs in polynomial time
- A problem is **#P-Hard** if given an O(1) oracle to solve it, every problem in #P can be solved in polynomial time (e.g. by reduction)
- A problem is **#P-Complete** if it belongs to both the classes #P and #P-Hard
- There are no known polytime algorithms for solving #P-Complete problems. If we found one it would imply that P = NP.

Examples of #P-Hard problems

- #SAT, i.e. how many satisfying solutions for a given SAT problem?
- How many solutions for a given DNF formula?
- How many solutions for a 2-SAT problem?
- How many perfect matchings for a bipartite graph?
- How many graph colorings (with k colors) for a given graph G?

Examples from https://en.wikipedia.org/wiki/%E2%99%AFP-complete

EXACT INFERENCE

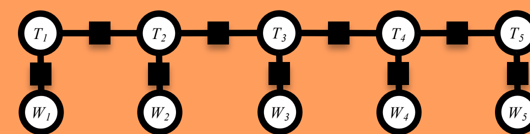
Exact Inference

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$



3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

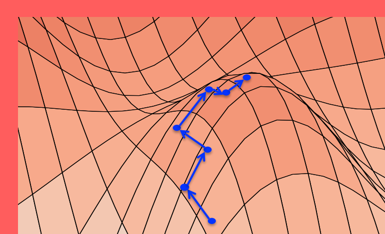
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

Three Tasks:

1. Marginal Inference (#P-Hard)
Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad \Bigg| \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function (#P-Hard)
Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference (NP-Hard)
Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} p(\boldsymbol{x} \mid \boldsymbol{\theta})$$