Statistical Methods for NLP

Language Models, Graphical Models

Sameer Maskey

Week 13, April 13, 2010

Some slides provided by Stanley Chen and from Bishop Book Resources

Announcements

- Final Project Due, April 20 (11:59pm)
  - Some requests to push the date
  - April 25 (11:59pm) (No extensions, no late days) – possibility?
- Project presentations April 27
  - Email me if you want to give the presentation in a particular slot
  - Randomized assignment
  - 10 min presentations (8 mins presentation) (2 mins Q&A)
- HW2 Returned
  - Homework was difficult
  - Average (75) → Highest (91)
  - Intermediate Report Returned
- Solutions for all available
  - Please come to my office hours to discuss them
  - For you to use the solutions for your other out of class project, will be available at the end of the semester

Final Report

- Maximum 8 pages (including references)
- Using ACL style
  - Latex or Word style
- Filename should be your UNI ID
- Needs to be pdf file
- Points will be taken off if any of the above requirements are not followed

Writing the Final Project Report

- You should clearly state the problem and your solution to the problem
- Related work
- Problems you faced and Implementation discussion
- Analysis and discussion of results
- Critical Analysis of why the solution works or why it does not? ← Important
- What changes you can suggest for future work

Topics for Today

- Language Models
- Recap: Bayesian Network
- Markov Random Fields

What's a Language Model?

- A language model is a probability distribution over word sequences
- p("nothing on the roof") $\approx$ 0.001
- p("huts sing on de woof") $\approx$ 0

Where Are Language Models Used?

- Speech recognition
- Handwriting recognition
- Spelling correction
- Optical character recognition
- Machine translation
- Natural language generation
- Information retrieval
- Any problem that involves sequences ?

Use of Language Model in Speech Recognition

$$
\begin{aligned}
W^* &= \underset{W}{\arg\max}\, P(W \mid X, \Theta) \\
&= \underset{W}{\arg\max}\, \frac{P(X \mid W, \Theta) P(W \mid \Theta)}{P(X)} && \text{Bayes' rule} \\
&= \underset{W}{\arg\max}\, P(X \mid W, \Theta) P(W \mid \Theta) && \text{P(X) doesn't depend on W}
\end{aligned}
$$

- W is a sequence of words, W* is the best sequence.
- X is a sequence of acoustic features.
- Θ is a set of model parameters.

Language Modeling and Domain

- Isolated digits: implicit language model

$$p(\text{"}one\text{"}) = \frac{1}{11}, p(\text{"}two\text{"}) = \frac{1}{11}, \ldots, p(\text{"}zero\text{"}) = \frac{1}{11}, p(\text{"}oh\text{"}) = \frac{1}{11}$$

- All other word sequences have probability zero
- Language models describe what word sequences the domain allows
- The better you can model acceptable/likely word sequences, or the fewer acceptable/likely word sequences in a domain, the better a bad acoustic model will look
- e.g. isolated digit recognition, yes/no recognition

N-gram Models

- It's hard to compute
  p("and nothing but the truth")
- Decomposition using conditional probabilities can help

p("and nothing but the truth") = p("and") x p("nothing"|"and") x p("but"|"and nothing") x p("the"|"and nothing but") x p("truth"|"and nothing but the")

The N-gram Approximation

- Q: What's a trigram? What's an n-gram?
  A: Sequence of 3 words. Sequence of n words.

- Assume that each word depends only on the previous two words (or n-1 words for n-grams)

p("and nothing but the truth") =	p("and") x p("nothing"|"and") x p("but"|"and nothing") x p("the"|"nothing but") x p("truth"|"but the")

- Trigram assumption is clearly false

  p(w | of the) vs. p(w | lord of the)
- Should we just make n larger?

  can run into data sparseness problem
- N-grams have been the workhorse of language modeling for ASR over the last 30 years
- Uses almost no linguistic knowledge

Technical Details: Sentence Begins & Ends

$$p(w = w_1 \ldots w_n) = \prod_{i=1}^{n} p(w_i \mid w_{i-2} w_{i-1})$$

Pad beginning with special beginning-of-sentence token:
$w_{-1} = w_0 = \triangleright$

Want to model the fact that the sentence is ending, so pad end with special end-of-sentence token:
$wn_{+1} = \triangleleft$

$$p(w = w_1 \ldots w_n) = \prod_{i=1}^{n+1} p(w_i \mid w_{i-2} w_{i-1})$$

Bigram Model Example

training data:

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

testing data / what's the probability of: JOHN READ A BOOK

$$p(JOHN \mid >) = \frac{count(> JOHN)}{count(>)} = \frac{1}{3}$$

$$p(READ \mid JOHN) = \frac{count(JOHN \cdot READ)}{count(JOHN)} = 1$$

$$p(A \mid READ) = \frac{count(READ \cdot A)}{count(READ)} = \frac{2}{3}$$

$$p(BOOK \mid A) = \frac{count(A \cdot BOOK)}{count(A)} = \frac{1}{2}$$

$$p(< \mid BOOK) = \frac{1}{2}$$

$$p(w) = \frac{1}{3} \cdot 1 \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{2}{36}$$

Trigrams, cont'd

Q: How do we estimate the probabilities?

A: Get real text, and start counting…

Maximum likelihood estimate would say:

p("the"|"nothing but") =
C("nothing but the") / C("nothing but")

where C is the count of that sequence in the data

Data Sparseness

- Let's say we estimate our language model from yesterday's court proceedings
- Then, to estimate
  p("to"|"I swear") we use
  count ("I swear to") / count ("I swear")
- What about p("to"|"I swerve") ?

  If no traffic incidents in yesterday's hearing,

  count("I swerve to") / count("I swerve")

  = 0 if the denominator > 0, or else is undefined

Very bad if today's case deals with a traffic incident!

Language Model Smoothing

- How can we adjust the ML estimates to account to account for the effects of the prior distribution when data is sparse?
- Generally, we don’t actually come up with explicit priors, but we use it as justification for *ad hoc* methods

Smoothing: Simple Attempts

- Add one: (V is vocabulary size)

$$p(z \mid xy) \approx \frac{C(xyz)+1}{C(xy)+V}$$

  Advantage: Simple
  Disadvantage: Works very badly

- What about delta smoothing:

$$p(z \mid xy) \approx \frac{C(xyz)+\delta}{C(xy)+V\delta}$$

A: Still bad…..

Smoothing: Good-Turing

- Basic idea: seeing something once is roughly the same as not seeing it at all
- Count the number of times you observe an event once; use this as an estimate for unseen events
- Distribute unseen events' probability equally over all unseen events
- Adjust all other estimates downward, so that the set of probabilities sums to 1
- Several versions; simplest is to scale ML estimate by (1-prob(unseen))

Good-Turing Example

- Imagine you are fishing in a pond containing {carp, cod, tuna, trout, salmon, eel, flounder, and bass}
- Imagine you've caught: 10 carp, 3 cod, 2 tuna, 1 trout, 1 salmon, and 1 eel so far.
- Q: How likely is it that the next catch is a new species (flounder or bass)?
- A: prob(new) = prob(1's) = 3/18
- Q: How likely is it that the next catch is a bass?
- A: prob(new)x0.5 = 3/36

Back Off

- (Katz, 1987) Use MLE if we have enough counts, otherwise *back off* to a lower-order model

$$p_{Katz}(w_i \mid w_{i-1}) = p_{MLE}(w_i \mid w_{i-1}) \quad \text{if count}(\mathrm{w_{i\text{-}1}w_i}) \geq 5$$

$$= p_{GT}(w_i \mid w_{i-1}) \quad \text{if } 1 \leq \text{count}(\mathrm{w_{i\text{-}1}w_i}) \leq 4$$

$$= \alpha_{w_{i-1}} p_{Katz}(w_i) \quad \text{if count}(\mathrm{w_{i\text{-}1}w_i}) = 0$$

- choose $\alpha_{w_{i-1}}$ so that $\sum_{w_i} p_{Katz}(w_i \mid w_{i-1}) = 1$

Smoothing: Interpolation

Idea: Trigram data is very sparse, noisy,
Bigram is less so,
Unigram is very well estimated from a large corpus
Interpolate among these to get the best combination

$$p(z \mid xy) = \lambda \frac{C(xyz)}{C(xy)} + \mu \frac{C(yz)}{C(y)} + (1 - \lambda - \mu) \frac{C(z)}{C(\bullet)}$$

Find $0 < \lambda , \mu < 1$ by optimizing on "held-out" data
Can use deleted interpolation in an HMM framework

Example

- Die  Possible outputs: 1,2,3,4,5,6
- Assume our training sequence is: x = 1,3,1,6,3,1,3,5
- Test sequence is: y = 5,1,3,4
- ML estimate from training:

$$\theta_m = (3/8, 0, 3/8, 0, 1/8, 1/8)$$

$$p_{\theta_m} (y) = 0$$

- Need to smooth θ_m

Example, cont'd

- Let $\theta_u = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$

- We can construct a linear combination from θ_m and θ_u

$$\theta_s = \lambda \theta_m + (1- \lambda) \theta_u \qquad 0 <= \lambda <= 1$$

- What should the value of $1- \lambda$ be?

- A reasonable choice is a/N, where a is a small number, and N is the training sample size

Example, cont'd

- e.g. if a=2, then $1-\lambda = 2/8 = 0.25$

$\theta_s = 0.75\ (.375, 0, .375, 0, .125, .125)$
$\quad + 0.25\ (.167, .167, .167, .167, .167, .167)$

$\quad = (.323, .042, .323, .042, .135, .135)$

Held-out Estimation

- Split training data into two parts:
  Part 1: $x_1^n = x_1 \ x_2 \ \ldots \ x_n$
  Part 2: $x_{n+1}^N = x_{n+1} \ x_{n+2} \ \ldots \ x_N$
- Estimate θ_m from part 1, combine with θ_u
  $\theta_s = \lambda \theta_m + (1-\lambda) \theta_u \quad 0 <= \lambda <= 1$
- Pick λ so as to maximize the probability of Part 2 of the training data
- Q: What if we use the **same** dataset to estimate the MLE estimate θ_m and λ?
  Hint: what does MLE stand for?

Smoothing: Kneser-Ney

- Combines back off and interpolation
- Motivation: consider bigram model
- Consider p(Francisco|eggplant)
- Assume that the bigram “eggplant Francisco” never occurred in our training data ... therefore we back off or interpolate with lower order (unigram) model
- Francisco is a common word, so both back off and interpolation methods will say it is likely
- But it only occurs in the context of “San” (in which case the bigram models it well)
- Key idea: Take the lower-order model to be the number of different contexts the word occurs in, rather than the unigram probability of the word

Smoothing: Kneser-Ney

- Subtract a constant D from all counts
- Interpolate against lower-order model which measures how many different contexts the word occurs in

- Modified K-N Smoothing: make D a function of the number of times the trigram xyz occurs

$$p(z \mid xy) = \frac{C(xyz) - D}{C(xy)} + \lambda \frac{C(\cdot z)}{\sum C(\cdot z)}$$

So, which technique to use?

- Empirically, interpolation is superior to back off
- State of the art is Modified Kneser-Ney smoothing (Chen & Goodman, 1999)

Does Smoothing Matter?

- No smoothing (MLE estimate):
  - Performance will be very poor
  - Zero probabilities will kill you
- Difference between bucketed linear interpolation (ok) and modified Kneser-Ney (best) is around 1% absolute in word error rate for a 3-gram model
- No downside to better smoothing (except in effort)
- Differences between best and suboptimal become larger as model order increases

Model Order

- Should we use big or small models?
  e.g. 3-gram or 5-gram?
- With smaller models, less sparse data issues → better probability estimates?
  - Empirically, bigger is better
  - With best smoothing, little or no performance degradation if model is too large
  - With lots of data (100M words +) significant gain from 5-gram
- Limiting resource: disk/memory
- Count cutoffs can be used to reduce the size of the LM
- Discard all n-grams with count less than threshold

Evaluating Language Models

- Best way: plug into your system (ASR, Summarizer, Text Categorizer), see how LM affects error rate
  - Expensive to compute
- Is there something cheaper that predicts WER well?
  - “perplexity” (PP) of test data (only needs text)
  - Doesn’t always predict WER well, but has theoretical significance
  - Predicts best when 2 LM’s being compared are trained on same data

Perplexity

- Perplexity is average branching factor, i.e. how many alternatives the LM believes there are following each word
- Another interpretation: $\log_2 PP$ is the average number of bits per word needed to encode the test data using the model P( )

- Ask a speech recognizer to recognize digits: 0,1,2,3,4,5,6,7,8,9 simple task (?) perplexity = 10
- Ask a speech recognizer to recognize alphabet: a,b,c,d,e,…z
  more complex task … perplexity = 26
- alpha, bravo, charlie … yankee, zulu
  perplexity = 26

Perplexity measures LM difficulty

Computing Perplexity

1. Compute the geometric average probability assigned to each word in test data $w_1..w_n$ by model P( )

$$p_{avg} = [\prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})]^{\frac{1}{n}}$$

2. Invert it: PP = $1/p_{avg}$

ML Models We Know in Graphical Representation

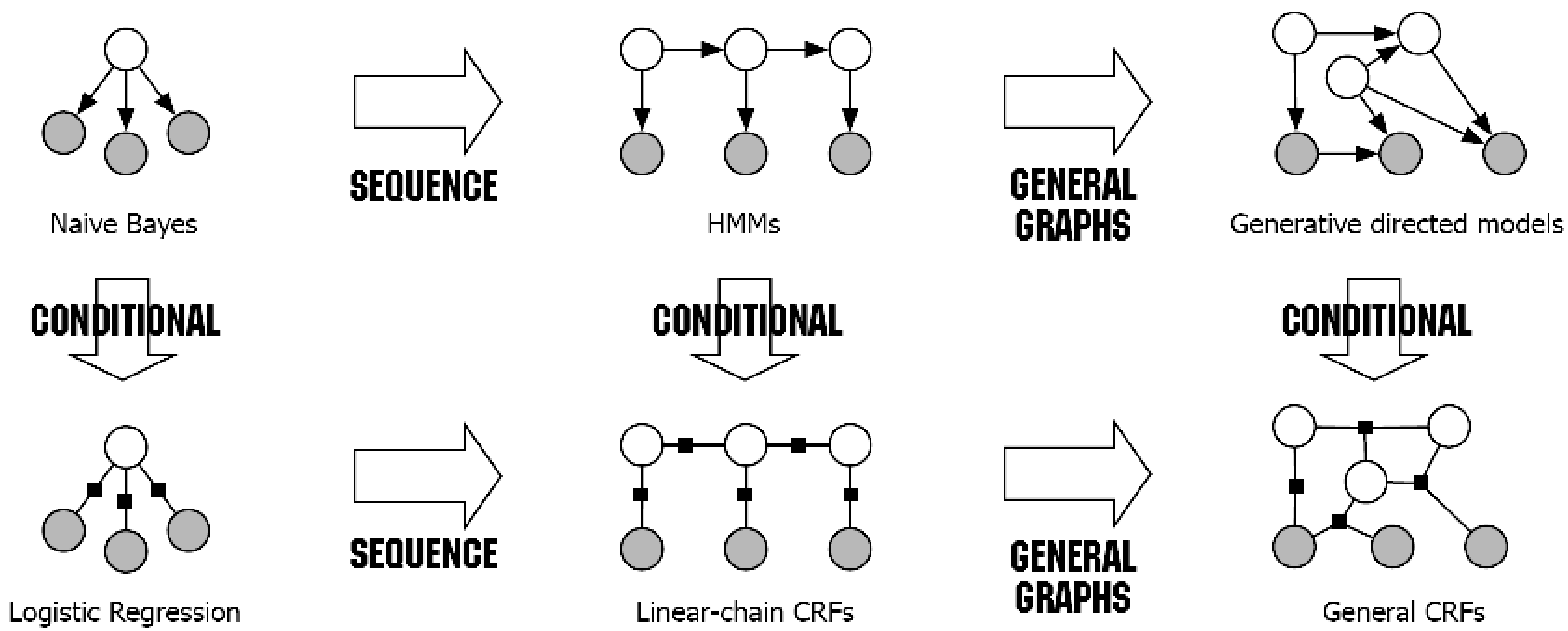
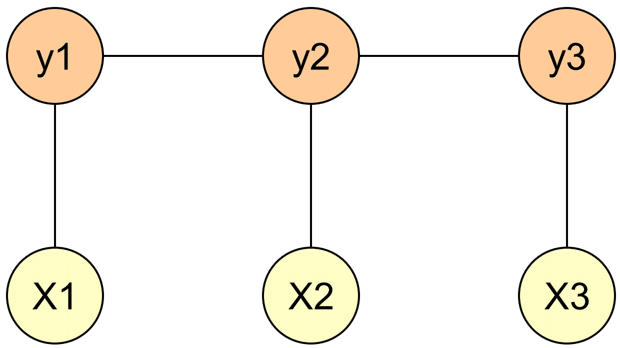
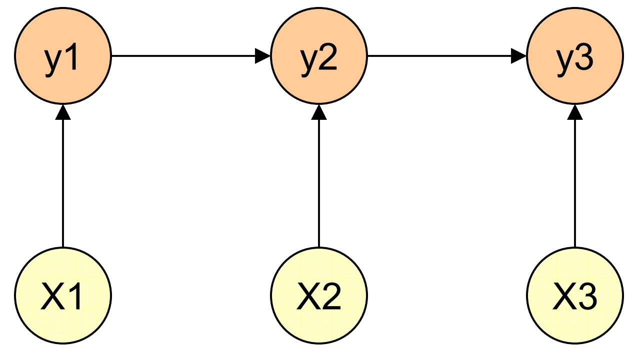


Figure from [1]

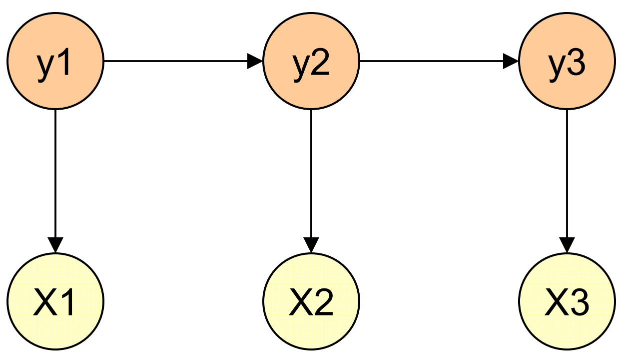
CRF, MEMM, HMM



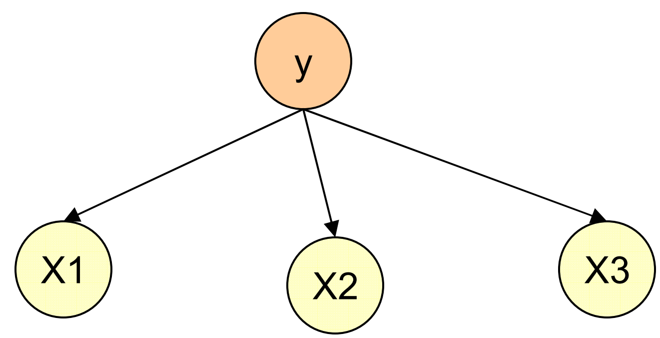
CRF



MEMM



HMM



Naïve Bayes

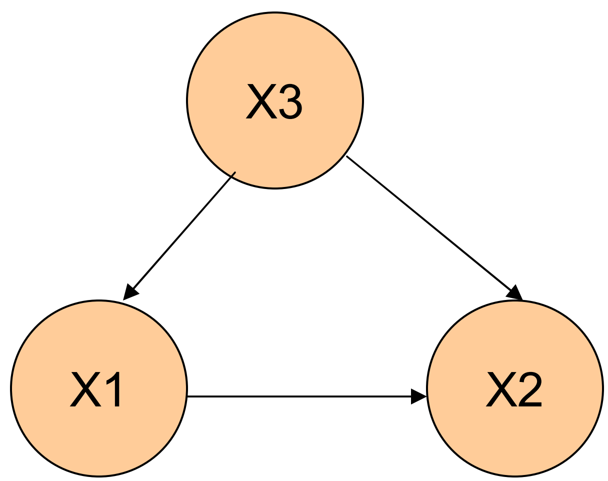
Review: Bayesian Networks

- Graph G where G is acyclic and is defined as follows

$$G = (V, E)$$

$$V = X_1, X_2, ..., X_N$$

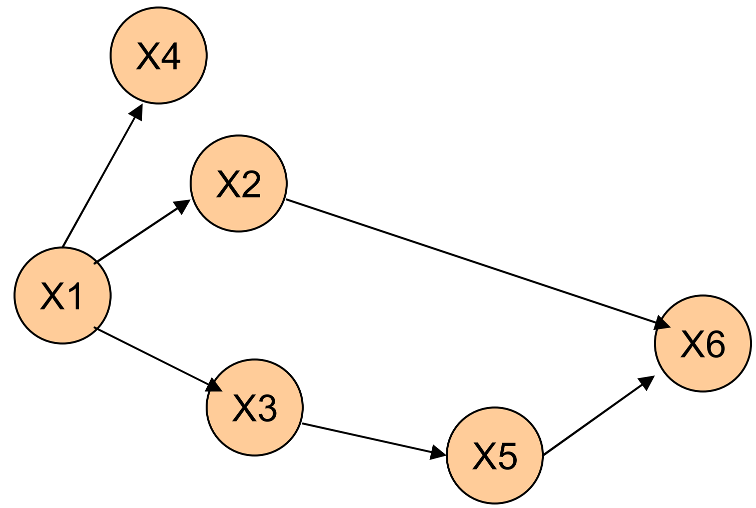
$$E = (X_i, X_j) : i \neq j$$



- Each node has a set of parents

Factorization of Joint Probability

- Factorization of joint probability reduces the number of parameters to estimate



$$P(x_1, ..., x_n) = \prod_{i=1}^{N} p(x_i|\pi_i)$$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) =$$

$\boxed{2^6}$

$$p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3)p(x_6|x_2, x_5)$$

$\boxed{2^1}$ $\boxed{2^2}$ $\boxed{2^2}$ $\boxed{2^2}$ $\boxed{2^2}$ $\boxed{2^3}$

- Conditional Probability Tables in each node are smaller

Conditional Independence

- a is independent of b given c

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

- Equivalently

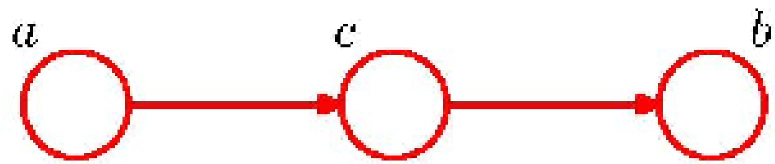
$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

- Notation

$$a \perp\!\!\!\perp b \mid c$$

**Slides from here forward are from Bishop Book Resource [3]*

Conditional Independence: Example 1

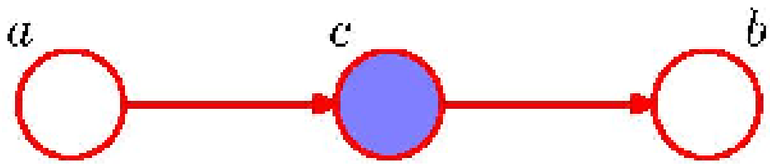


$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a) \sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

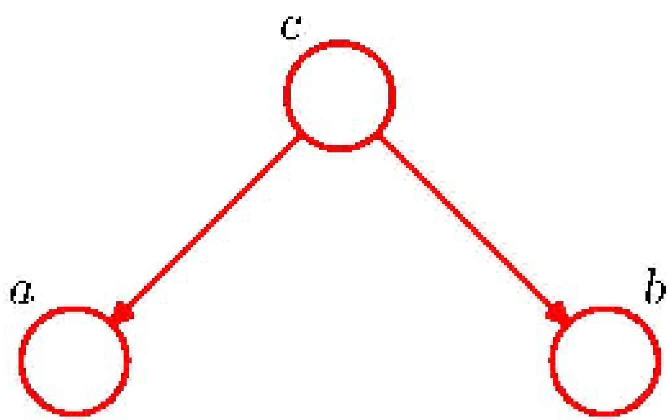
Conditional Independence: Example 1



$$
\begin{aligned}
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\
&= p(a|c)p(b|c)
\end{aligned}
$$

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 2

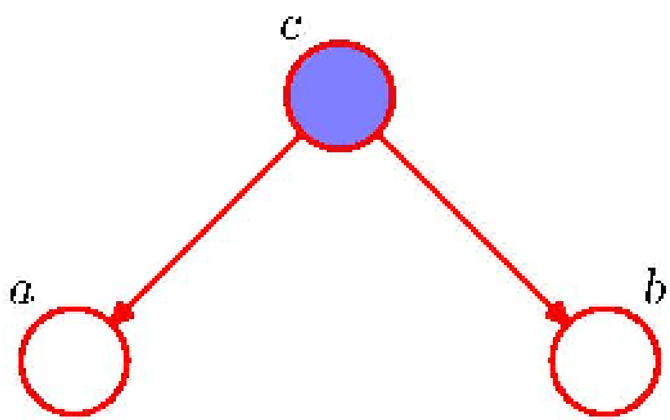


$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

$$p(a, b) = \sum_c p(a|c)p(b|c)p(c)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

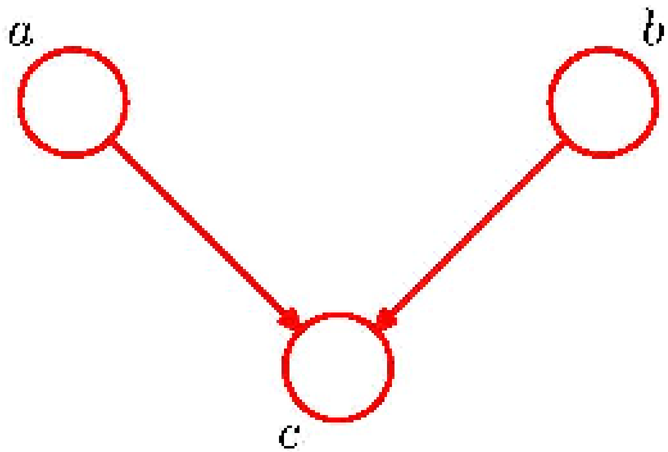
Conditional Independence: Example 2



$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = p(a|c)p(b|c)$$

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 3



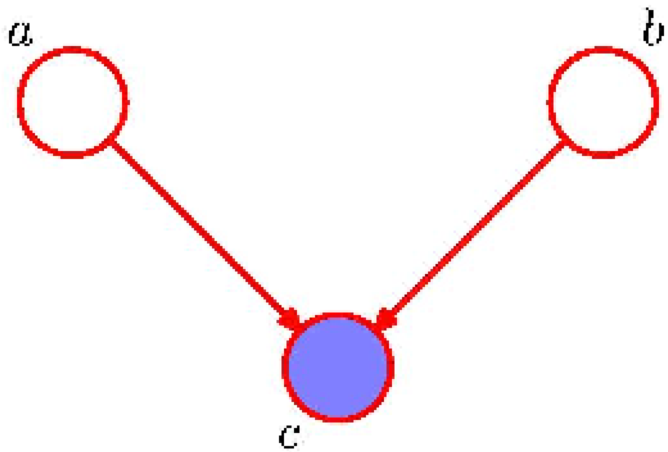
$$p(a, b, c) = p(a)p(b)p(c|a, b)$$

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

$$a \perp\!\!\!\perp b \mid \emptyset$$

- Note: this is the opposite of Example 1, with c unobserved.

Conditional Independence: Example 3



$$p(a, b|c) = \frac{p(a, b, c)}{p(c)}$$

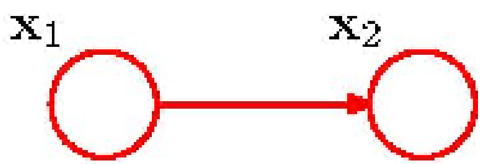
$$= \frac{p(a)p(b)p(c|a, b)}{p(c)}$$

$$a \not\perp\!\!\!\perp b \mid c$$

Note: this is the opposite of Example 1, with c observed.

Joint Distribution

- General joint distribution: $K^2 - 1$ parameters



$$p(\mathbf{x}_1, \mathbf{x}_2|\boldsymbol{\mu}) = \prod_{k=1}^{K} \prod_{l=1}^{K} \mu_{kl}^{x_{1k} x_{2l}}$$

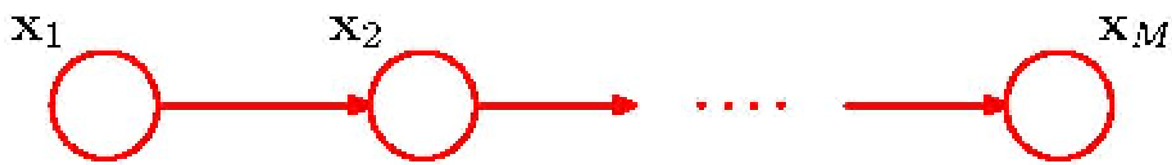
- Independent joint distribution: $2(K - 1)$ parameters

$$\hat{p}(\mathbf{x}_1, \mathbf{x}_2|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_{1k}^{x_{1k}} \prod_{l=1}^{K} \mu_{2l}^{x_{2l}}$$

Discrete Variables (2)

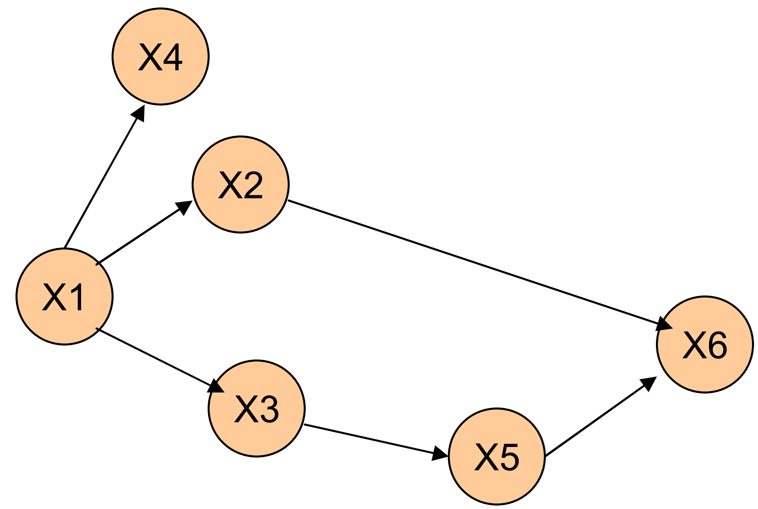
General joint distribution over M variables:
$K^M - 1$ parameters

M-node Markov chain: $K - 1 + (M-1)K(K-1)$ parameters



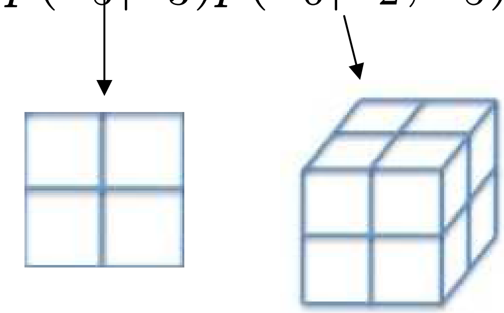
Conditional Probability Tables

$$P(x_1, \dots, x_n) = \prod_{i=1}^{N} p(x_i|\pi_i)$$



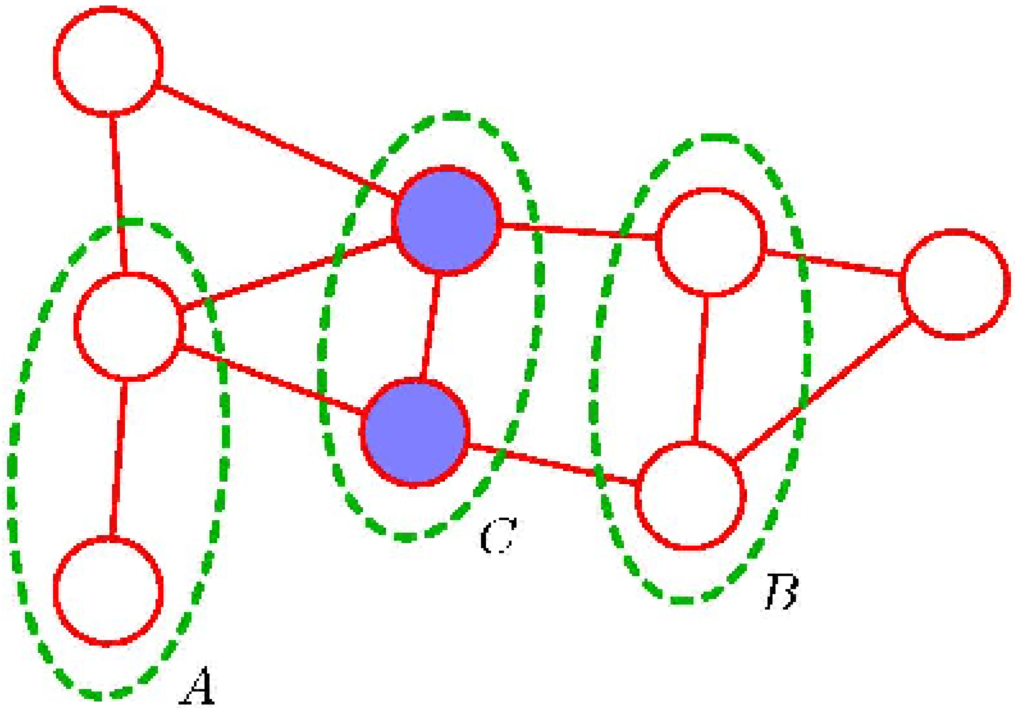
$$p(x_1, x_2, x_3, x_4, x_5, x_6) =$$

$$p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3)p(x_6|x_2, x_5)$$



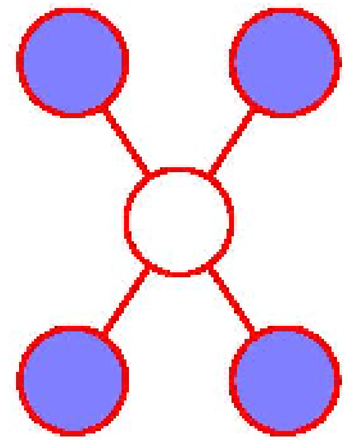
$$\theta(x_i, \pi_i) = \frac{m(x_i, \pi_i)}{m(\pi_i)}$$

Markov Random Fields



$$A \perp\!\!\!\perp B \mid C$$

Markov Blanket



References

- [1] Sutton, C. and McCallum, A., "An Introduction to Conditional Random Fields for Relational Learning" 2006
- [2] Jurafsky, D and Martin, J, “Speech and Language Processing,” 2009
- [3] Christopher Bishop, “Pattern Recognition and Machine Learning” 2006