Graphical Models

CPS 570

Ron Parr

Why Joint Distributions are Important

- Joint distributions gives $P(X_1 \ldots X_n)$
- Classification/Diagnosis
  - Suppose X1=disease
  - X2…Xn = symptoms
- Co-occurrence
  - Suppose X3=lung cancer
  - X5=smoking
- Rare event Detection
  - Suppose X1…Xn = parameters of a credit card transaction
  - Call card holder if P(X1…Xn) is below threshold?

Modeling Joint Distributions

- To do this correctly, we need a full assignment of probabilities to all atomic events
- Unwieldy in general for discrete variables: n binary variables = 2^n atomic events
- Independence makes this tractable, but too strong (rarely holds)
- Conditional independence is a good compromise: Weaker than independence, but still has great potential to simplify things

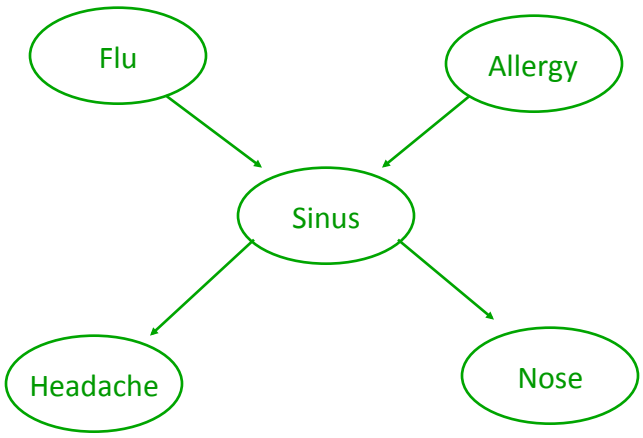
Overview

- Conditional independence
- Bayesian networks
- Variable Elimination
- Sampling
- Factor graphs
- Belief propagation
- Undirected models

Conditional Independence

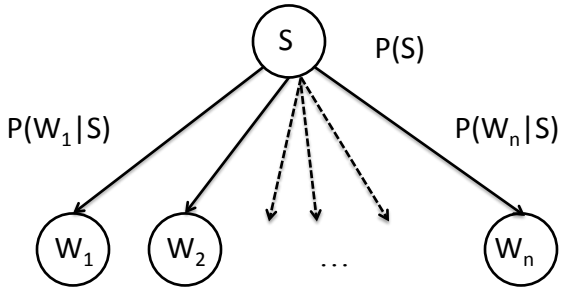
- Suppose we know the following:
  - The flu causes sinus inflammation
  - Allergies cause sinus inflammation
  - Sinus inflammation causes a runny nose
  - Sinus inflammation causes headaches
- How are these connected?

Example 1: Simple graphical structure



Knowing sinus separates the variables from each other.

Example 2: Naïve Bayes Spam Filter



We will see later why this is a particularly convenient representation. (Does it make a correct assumption?)

Conditional Independence

- We say that two variables, A and B, are conditionally independent given C if:
  - P(A|BC) = P(A|C)
  - P(AB|C) = P(A|C)P(B|C)
- How does this help?
- We store only a conditional probability table (CPT) of each variable given its parents
- Naïve Bayes (e.g. Spam Assassin) is a special case of this!

Notation Reminder

- P(A|B) is a conditional prob. distribution
  - It is a function!
  - P(A=true|B=true), P(A=true|B=false), P(A=false|B=True), P(A=false|B=true)
- P(A|b) is a probability distribution, function
- P(a|B) is a function, not a distribution
- P(a|b) is a number

What is Bayes Net?

- A directed acyclic graph (DAG)
- Given parents, each variable is *independent of non-descendants*
- Joint probability decomposes:

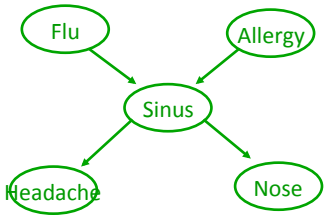
$$P(x_1 .. x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$$

- For each node X_i, store $P(X_i|\text{parents}(X_i))$
- Call this a Conditional Probability Table (CPT)
- CPT size is exponential in number of parents

Real Applications of Bayes Nets

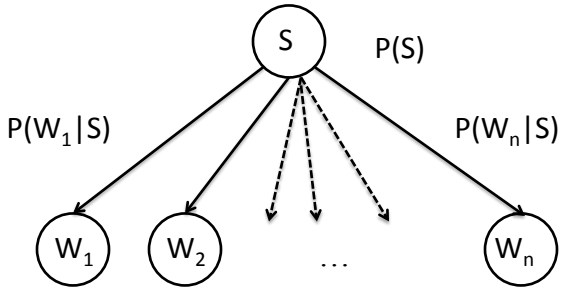
- Diagnosis of lymph node disease
- Used in Microsoft office and Windows
- Used by robots to identify meteorites to study
- Study the human genome: Alex Hartemink et al.
- Many other applications…

Space Efficiency



- Entire joint distribution as 32 (31) entries
  - P(H|S),P(N|S) have 4 (2)
  - P(S|AF) has 8 (4)
  - P(A) has 2 (1)
  - Total is 20 (10)
- This can require exponentially less space
- Space problem is solved for "most" problems

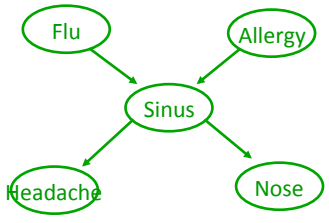
Naïve Bayes Space Efficiency



Entire Joint distribution has 2^{n+1} (2^{n+1}-1) numbers vs. 4n+2 (2n+1)

Atomic Event Probabilities

$$P(x_1..x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$$



Note that this is guaranteed true if we construct net incrementally, so that for each new variable added, we connect all influencing variables as parents (prove it by induction)

(Non)Uniqueness of Bayes Nets

- You can always construct a valid Bayes net by inserting variables one at a time
- Order of adding variables can lead to different Bayesian networks for the same distribution
- Suppose A and B are independent, but C is a function of A and B
  - Add A, B, then C:
  - Add C, A, then B:

Doing Things the Hard Way

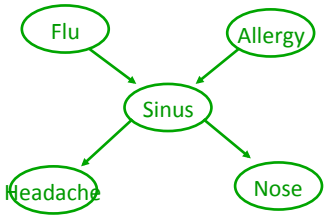
$$P(f \mid h) = \frac{P(fh)}{P(h)} = \frac{\sum_{SAN} P(fhSAN)}{\sum_{SANF} P(hSANF)} = \frac{\sum_{SAN} P(f)P(A)P(S \mid Af)P(h \mid S)P(N \mid S)}{\sum_{SANF} P(F)P(A)P(S \mid AF)P(h \mid S)P(N \mid S)}$$

defn. of conditional probability

$$P(hSANF) = \prod_x p(x \mid parents(x))$$
$$= P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)$$

Doing this naïvely, we need to sum over all atomic events defined over these variables. There are exponentially many of these.

Working Smarter



$$P(h) = \sum_{SANF} P(hSANF)$$
$$= \sum_{SANF} P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)$$
$$= \sum_{NS} P(h \mid S)P(N \mid S) \sum_{AF} P(S \mid AF)P(A)P(F)$$
$$= \sum_{S} P(h \mid S) \sum_{N} P(N \mid S) \sum_{AF} P(S \mid AF)P(A)P(F)$$

Potential for exponential reduction in computation.

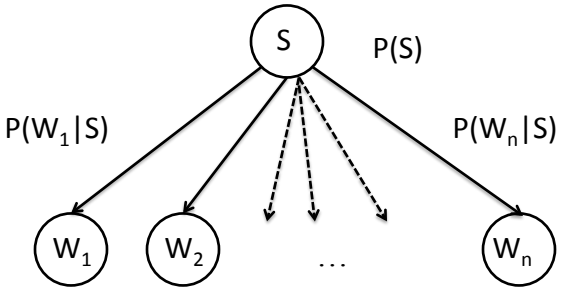
Computational Efficiency

$$\sum_{SANF} P(hSANF) = \sum_{SANF} P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)$$
$$= \sum_{S} P(h \mid S) \sum_{N} P(N \mid S) \sum_{AF} P(S \mid AF)P(A)P(F)$$

The distributive law allows us to decompose the sum.
AKA: Variable elimination
Analogous to Gaussian elimination but exponentially more expensive

Potential for an exponential reduction in computation costs.

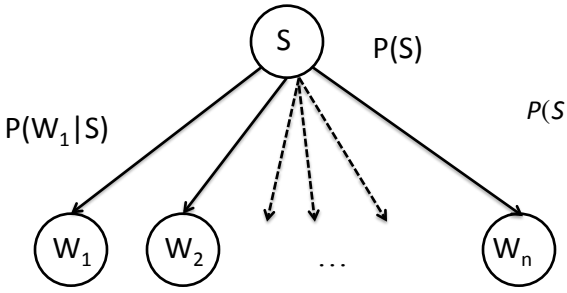
Naïve Bayes Efficiency



Given a set of words, we want to know which is larger: $P(s|W_1 \ldots W_n)$ or $P(\neg s|W_1 \ldots W_n)$.

Use Bayes Rule: $P(S \mid W_1 \ldots W_n) = \frac{P(W_1 \ldots W_n \mid S)P(S)}{P(W_1 \ldots W_n)}$

Naïve Bayes Efficiency II



$P(S \mid W_1 \ldots W_n) = \frac{P(W_1 \ldots W_n \mid S)P(S)}{P(W_1 \ldots W_n)}$

Observation 1: We can ignore $P(W_1 \ldots W_n)$
Observation 2: P(S) is given
Observation 3: $P(W_1 \ldots W_n|S)$ is easy: $P(W_1 \ldots W_n \mid S) = \prod_{i=1}^{n} P(W_i \mid S)$

Note: Can also do variable elimination by summing out leaves first.

Checkpoint

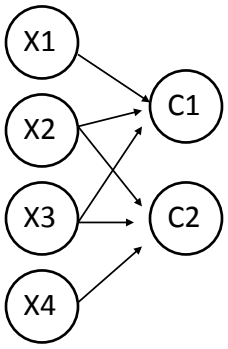
- BNs can give us an exponential reduction in the space required to represent a joint distribution.
- Storage is exponential in largest parent set.
- Claim: Parent sets are often reasonable.
- Claim: Inference cost is often reasonable.
- Question: Can we quantify relationship between structure and inference cost?

Now the Bad News…

- In full generality: Inference is NP-hard
- Decision problem: Is P(X)>0?
- We reduce from 3SAT
- 3SAT variables map to BN variables
- Clauses become variables with the corresponding SAT variables as parents

Reduction

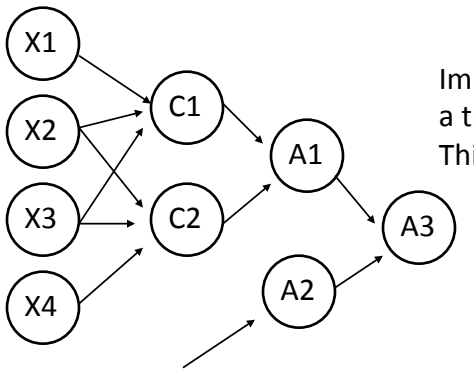
$$(\overline{X}_1 \vee X_2 \vee X_3) \wedge (\overline{X}_2 \vee X_3 \vee X_4) \wedge \ldots$$



Problem: What if we have a large number of clauses? How does this fit into our decision problem framework?

And Trees

We could make a single variable which is the AND of all of our clauses, but this would have CPT that is exponential in the number of clauses.



Implement as a tree of ANDs. This is polynomial.

Is BN Inference NP Complete?

- Can show that BN inference is #P hard
- #P is counting the number of satisfying assignments

- Idea: Assign variables uniform probability
- Probability of conjunction of clauses tells us how many assignments are satisfying

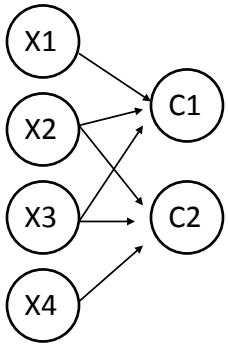
Checkpoint

- BNs can be very compact
- Worst case: Inference is intractable

- Hope that worst is case:
  - Avoidable (frequently, but no free lunch)
  - Easily characterized in some way

(Undirected) Trees

- Are the structures from our reduction trees?



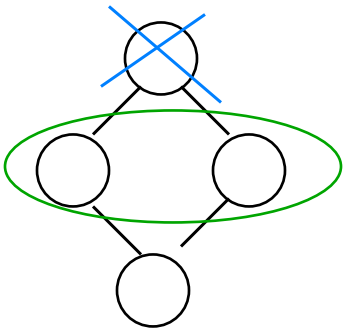
No!

- They are a Directed Acyclic Graph (DAG)
- BNs are always DAGs (sometimes trees)

Clues in the Graphical Structure

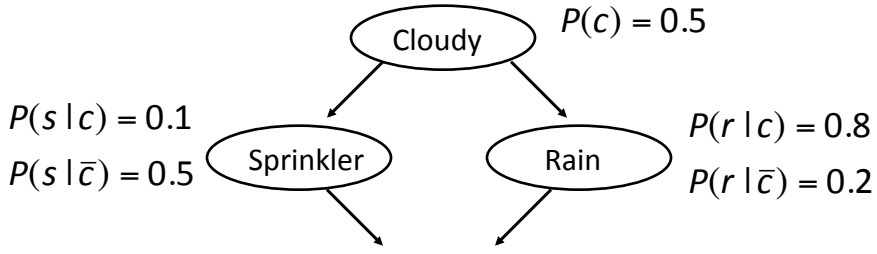
- Q: How does graphical structure relate to our ability to push in summations over variables?
- A:
  - We relate summations to graph operations
  - Summing out a variable =
    - Removing node(s) from DAG
    - Creating new replacement node
  - Relate graph properties to computational efficiency

Variable Elimination as Graph Operations



We can think of summing out a variable as creating a new "super variable" which contains all of that variable's neighbors

Another Example Network



$P(w \mid sr) = 0.99$

$P(w \mid s\bar{r}) = 0.9$

$P(w \mid \bar{s}r) = 0.9$

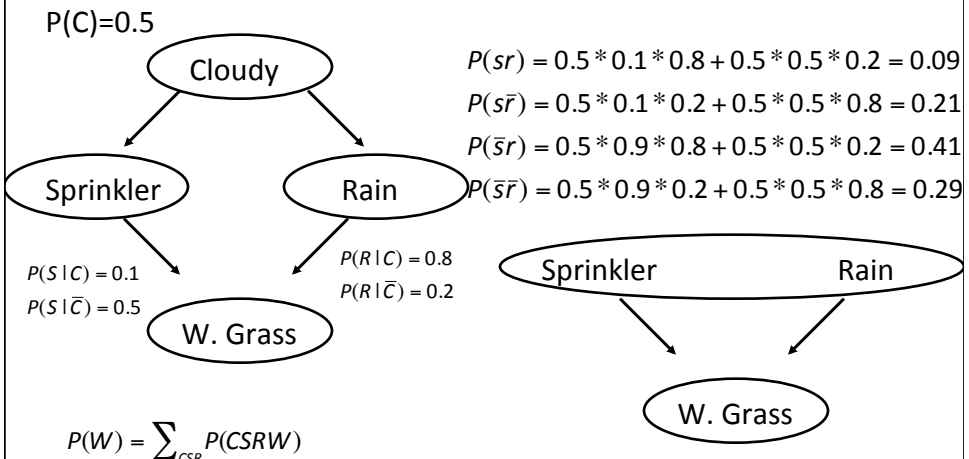
$P(w \mid \bar{s}\bar{r}) = 0.0$

Marginal Probabilities

Suppose we want P(W):

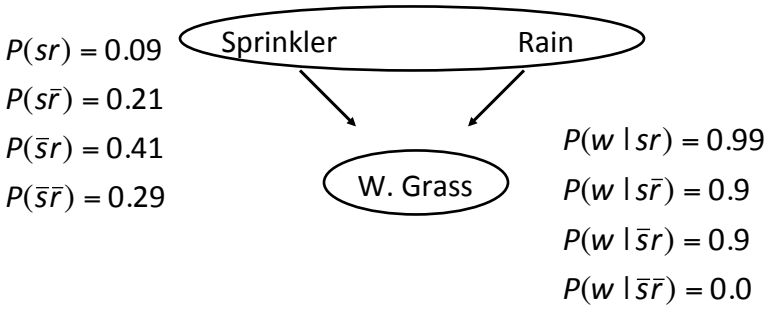
$$P(W) = \sum_{CSR} P(CSRW)$$
$$= \sum_{CSR} P(C)P(S \mid C)P(R \mid C)P(W \mid RS)$$
$$= \sum_{SR} P(W \mid RS) \sum_{C} P(S \mid C)P(C)P(R \mid C)$$

Eliminating Cloudy



$$P(W) = \sum_{CSR} P(CSRW)$$
$$= \sum_{CSR} P(C)P(S \mid C)P(R \mid C)P(W \mid RS)$$
$$= \sum_{SR} P(W \mid RS) \sum_{C} P(S \mid C)P(C)P(R \mid C)$$

Eliminating Sprinkler/Rain



$$P(w) = \sum_{SR} P(w \mid RS)P(RS)$$
$$= 0.09 * 0.99 + 0.21 * 0.9 + 0.41 * 0.9 + 0.29 * 0$$
$$= 0.6471$$

Dealing With Evidence

Suppose we have observed that the grass is wet?
What is the probability that it has rained?

$$P(R \mid W) = \alpha P(RW)$$
$$= \alpha \sum_{CS} P(CSRW)$$
$$= \alpha \sum_{CS} P(C)P(S \mid C)P(R \mid C)P(W \mid RS)$$
$$= \alpha \sum_{C} P(R \mid C)P(C) \sum_{S} P(S \mid C)P(W \mid RS)$$

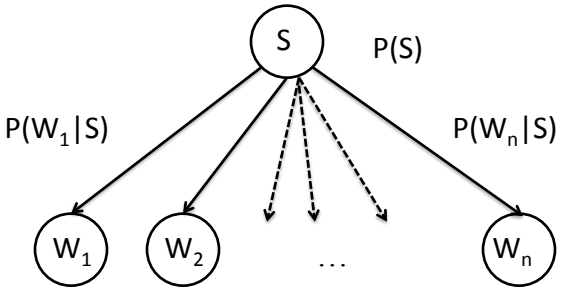
Is there a more clever way to deal with w?
Only keep the relevant parts.

Efficiency of Variable Elimination

- Exponential in the largest domain size of new variables created (just as in CSPs)
- Equivalently: Exponential in largest function created by pushing in summations (sum-product algorithm)

- Linear for trees
- Almost linear for almost trees ☺

Naïve Bayes Efficiency



Another way to understand why Naïve Bayes is efficient:
It's a tree!

Facts About Variable Elimination

- Picking variables in optimal order is NP hard
- For some networks, there will be no elimination ordering that results in a poly time solution
  (Must be the case unless P=NP)
- Polynomial for trees
- Need to get a little fancier if there are a large number of query variables or evidence variables

Sampling

- A Bayes net is an example of a **generative model** of a probability distribution
- Generative models allow one to generate samples from a distribution in a natural way
- Sampling algorithm:
  - While some variables are not sampled
    - Pick variable x with no unsampled parents
    - Assign this variable a value from p(x|parents(x))
  - Do this n times
  - Compute P(a) by counting in what fraction a is true

Comments on Sampling

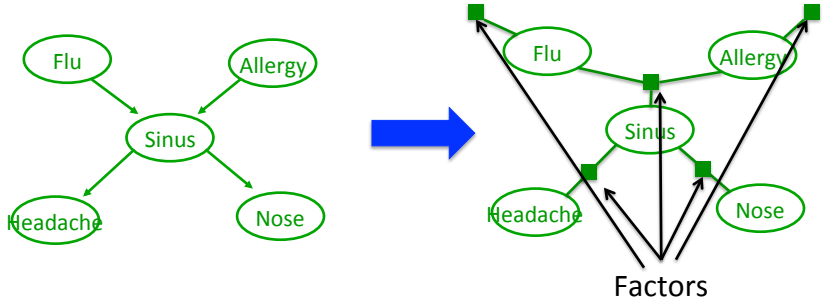
- Sampling is the easiest algorithm to implement
- Can compute marginal or conditional distributions by counting
- Not efficient in general

- Problem: How do we handle observed values?
  - Rejection sampling: Quit and start over when mismatches occur
  - Importance sampling: Use a reweighting trick to compensate for mismatches

- More clever approaches to sampling are possible

Factor Graphs

- Factor graph makes CPTs objects graph
- Need two types of nodes: vars and factors



Properties of Factor Graphs

- Always bipartite
- Can combine (multiply together) factors with nodes that are subsets of each other
  - Prior probabilities for parentless nodes can get combined with their downstream CPTs
  - Flu is multiplied in with P(S|F)
  - Allergy is multiplied in with P(A|F)
- In general, we create one factor for each maximal clique in a graph

Belief Propagation

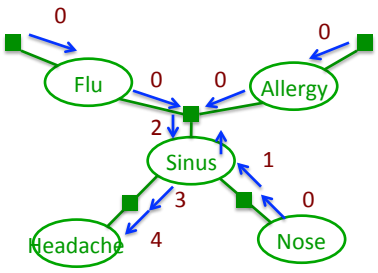
- BP passes "messages" on a factor graph
- Assume for now that our graph is a tree
- Impose an ordering (make one node root)
- BP is two-pass algorithm
- Pass messages from leaves to root (forward)
- Pass messages from root to leaves (backward)

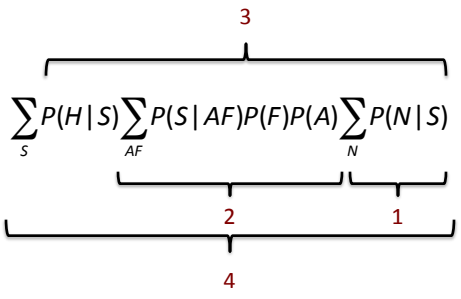
Messages: Forward Pass

- Variable nodes multiply together their upstream factor nodes and pass the resulting function down
- Factor nodes multiply together their upstream variable node messages with factor, sum out all but the upstream variable, pass the resulting function up

- If neighbor set is empty, just pass ones

What does the forward pass compute?





Forward pass is the same as doing variable elimination to the root!

Backward Pass

- Backward pass repeats forward steps, but in the opposite direction
- After forward and backward steps are complete, for any variable x is equal to the product of the incoming messages (from both passes)
- Why is this true?

- Insight: BP computes marginal probabilities of all nodes in 2X the cost of variable elimination for a single node

Generalizing BP

- Can compute probabilities of sets of variables that appear in a common factor by multiplying together the factor with all incoming messages (and possibly marginalizing the variables we don't care about)

- What if we don't have a tree?

"Loopy" BP

- What happens if we run BP on a graph that isn't a tree?
  - If we run BP multiple times on a tree, nothing changes
  - If we run BP multiple times on graph, messages can change due to loops
- Does this converge? (sometimes, yes)
- Does it give a reasonable answer (often, yes)

Maximizing

- Setting of variables with the highest probability?
- Can distribute max, just like product:

$$\sum_{S} P(H \mid S) \sum_{AF} P(S \mid AF) P(F) P(A) \sum_{N} P(N \mid S)$$

$$\max_{HS} P(H \mid S) \max_{AF} P(S \mid AF) P(F) P(A) \max_{N} P(N \mid S)$$

- Note: This only gives us the probability of the highest prob assignment; need work backwards to recover the actual assignment
- Can be used with loopy BP too

Summary of Algorithms for BNs

- Enumeration (consider all atomic events)
  - Exponential
  - Yuck!
- Variable elimination
  - Can be dramatically more efficient than enumeration
  - No guarantee of polynomial time
- Sampling
  - Easy to implement
  - May converge slowly in practice
- Belief Propagation
  - Exact for trees
  - "Loopy" for graphs
- Can modify algorithms to compute max prob. assignments

Other Algorithms

- Exact:
  - Can "compile" a non-tree BN into an equivalent factor graph by clustering variables
  - No free lunch: Factors may be very large
- Approximate:
  - Can split up large clusters
  - "mini-bucket" or variational approximations

Bayes Net Summary

- Bayes net = data structure for joint distribution
- Can give exponential reduction in storage
- Variation elimination and variants for tree-ish networks:
  - simple, elegant methods
  - efficient for many networks
- For some networks, must use approximation

- BNs are a major success story for modern AI
  - BNs do the "right" thing (no ugly approximations)
  - Exploit structure in problem to reduce storage/computation
  - Not always efficient, but inefficient cases are well understood
  - Work and used in practice

Undirected Models

- BNs are great, but…
  - Determining conditional independence is tricky
  - "D-separation" = a complicated set of rules for determining which variables depend upon which others given evidence

- Markov Random Fields (MRFs), also known as Markov Networks
  - Have all undirected arcs
  - Have more salient dependency relationships
  - Are (in some ways) easier to specify

MRF Specifications

- An MRF is a graph w/one node per random variable
- An MRF defines a function called a potential j(C) for each maximal clique in the graph
- What's a maximal clique:
  - A clique is a fully connected set of nodes
  - A maximal clique is not part of any larger clique

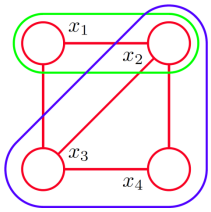


Figure © Chris Bishop

Probabilities for MRFs

$$p(\mathbf{x}) = \frac{1}{Z}\prod_C \varphi_C(\mathbf{x}_C)$$

This is very compact notation; parse it carefully!

$$z = \sum_{\mathbf{x}}\prod_C \varphi_C(\mathbf{x}_C)$$

- **x** is an assignment to all RVs in the network
- $\mathbf{x}_C$ is the subset of the variables in clique C
- z is a normalizing constant
- Good news: Potentials don't need to be distributions
- Bad news: Hard to compute z

Image De-Noising Example

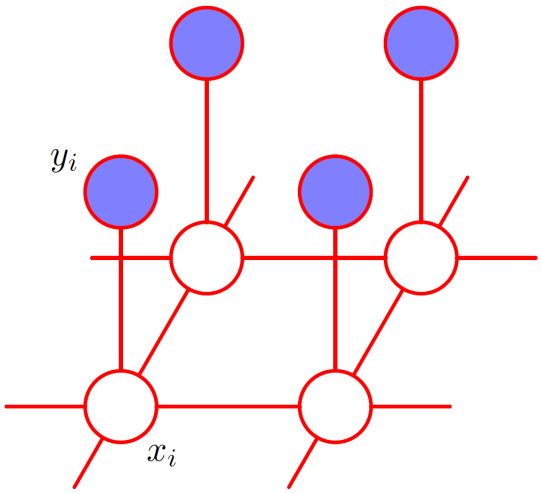
- Suppose we observe a 2-color digital image with pixels values y_i that are corrupted by some noise
- Suppose that the true image has pixel values x_i
- Assumptions:
  - y_i and x_i should be strongly correlated
  - x_i and x_j should be strongly correlated if i and j are neighbors

Image De-Noising Applied

Original Corrupted De-noised

Figure © Chris Bishop

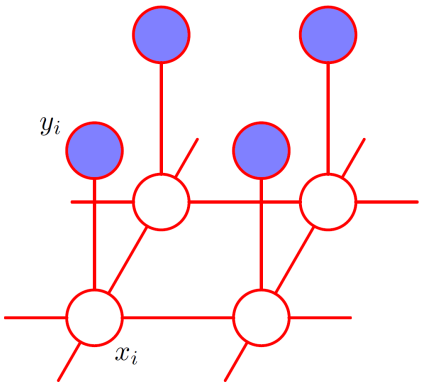
MRF Structure



Note: Simple structure means that cliques are defined just over pairs

Figure © Chris Bishop

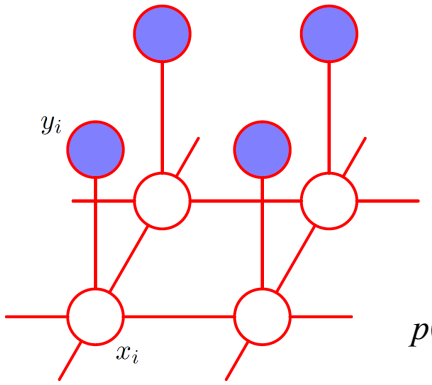
Potentials



$$\varphi(x_i) = e^{hx_i}$$
$$\varphi(x_i, x_j) = e^{-\beta x_i x_j}$$
$$\varphi(x_i, y_i) = e^{-\eta x_i y_i}$$

Note: x_i is not a maximal clique, but this is just a mater of convenience since there is an equivalent formulation in terms of maximal

Probability of an Assignment



$$p(x,y) = \frac{1}{z}\prod_i e^{hx_i}\prod_{i,j} e^{-\beta x_i x_j}\prod_i e^{-\eta x_i y_i}$$

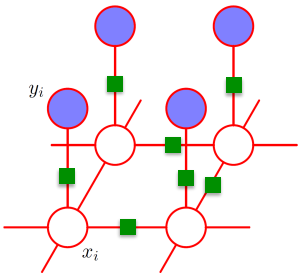
How do we deal with this?

Dealing with MRF Potentials

- We could use something like variable elimination to compute Z, but would be brutally expensive for networks like our image network
- Do we really need p(x,y)?
  - Marginal probabilities of nodes may suffice
  - Most likely assignment may suffice

Markov Networks and Factor Graphs

- Convert Markov networks to factor graphs



- Apply loopy BP to the factor graph
- Node marginals will be unnormalized

Sampling

- How do we sample from an undirected network (which node to sample first?)
- For i in nodes
  - Freeze the value of i's neighbors
  - Sample value of node i conditioned on value of i's neighbors (multiply factors to get the joint distribution; conditional is derived from the joint)

Sampling II

- Our simple sampling algorithm will, if run long enough, produce samples from the distribution implied by the Markov network
- How to use this:
  - Run the sampling algorithm "long enough"
  - Record all node values as a single sample
  - Repeat

Sampling III

- Our sampling algorithm is a specific instance of a general technique called "Gibbs Sampling"
- Gibbs sampling is a specific instance of a general technique called Markov Chain Monte Carlo (MCMC) sampling
- Can be applied to:
  - Bayes nets (though you need to be careful about what "neighbor" means
  - Arbitrary probability distributions if you are careful about how you set up your sampling scheme

Summary of Undirected Networks

- Undirected networks take use purely local information about relationships between variables to defined a joint probability distribution
- More natural for problems (e.g., images)
- Normalization constant is problematic
- Approximate inference techniques (loopy BP, sampling) often used

Summary of Graphical Models

- Powerful language for expressing relationships between variables w/o exponential cost of expressing the joint distribution
- Some network structures can be dealt with tractably (trees, and things close to trees)
- Approximate methods used for nastier networks