Machine Learning and Graphical Models

(Lecture I)

王立威

北京大学

信息科学技术学院

http://www.cis.pku.edu.cn/faculty/vision/wangliwei/

wanglw@cis.pku.edu.cn

Outline

- A brief overview of Machine Learning
- Graphical Models
  - Representation
  - Inference
  - Learning

- Definition of Machine Learning:
  - Learning from experiences.

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

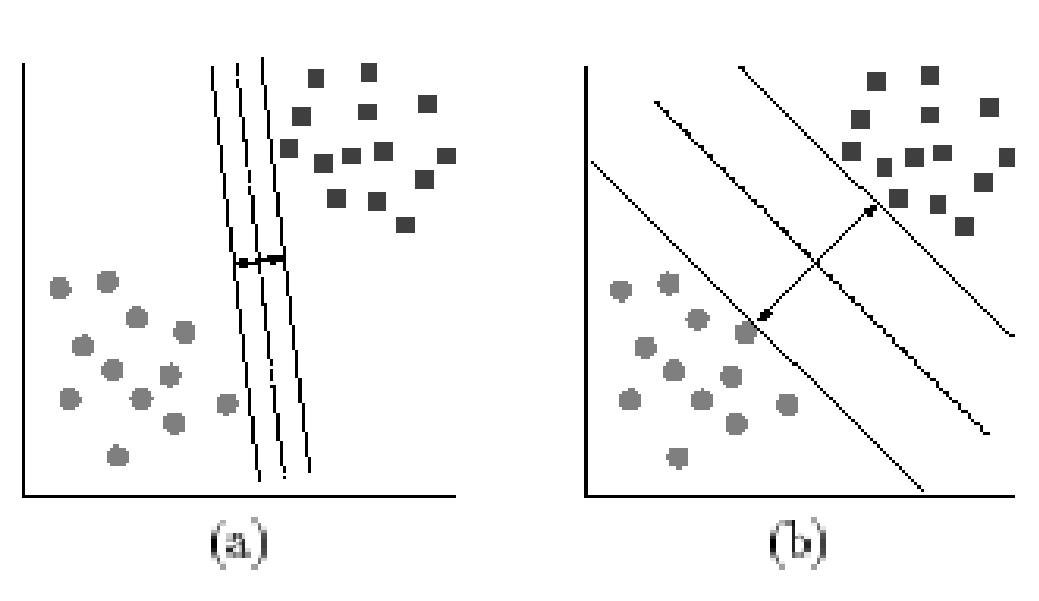
- Tom Mitchell

- "Classical" Machine Learning Tasks:
  - Classification: $f : \mathbf{R}^n \to \{-1,1\}$
    - spam filter, face recognition, …
  - Regression $f : \mathbf{R}^n \to \mathbf{R}$
    - Hook's law, Kepler's law,…
  - Ranking $f : \mathbf{R}^n \to \mathbf{R}$
    - Search engine
  - Probability (Distribution) Estimation

- "Classical" Machine Learning Algorithms
  - Classification
    - SVM
    - Boosting
    - Random Forest
    - Bagging
    - (Deep) Neural Networks
  - Regression
    - Lasso
    - Boosting

Support Vector Machines (SVMs)

- SVM: the large l_2 / l_2 margin classifier



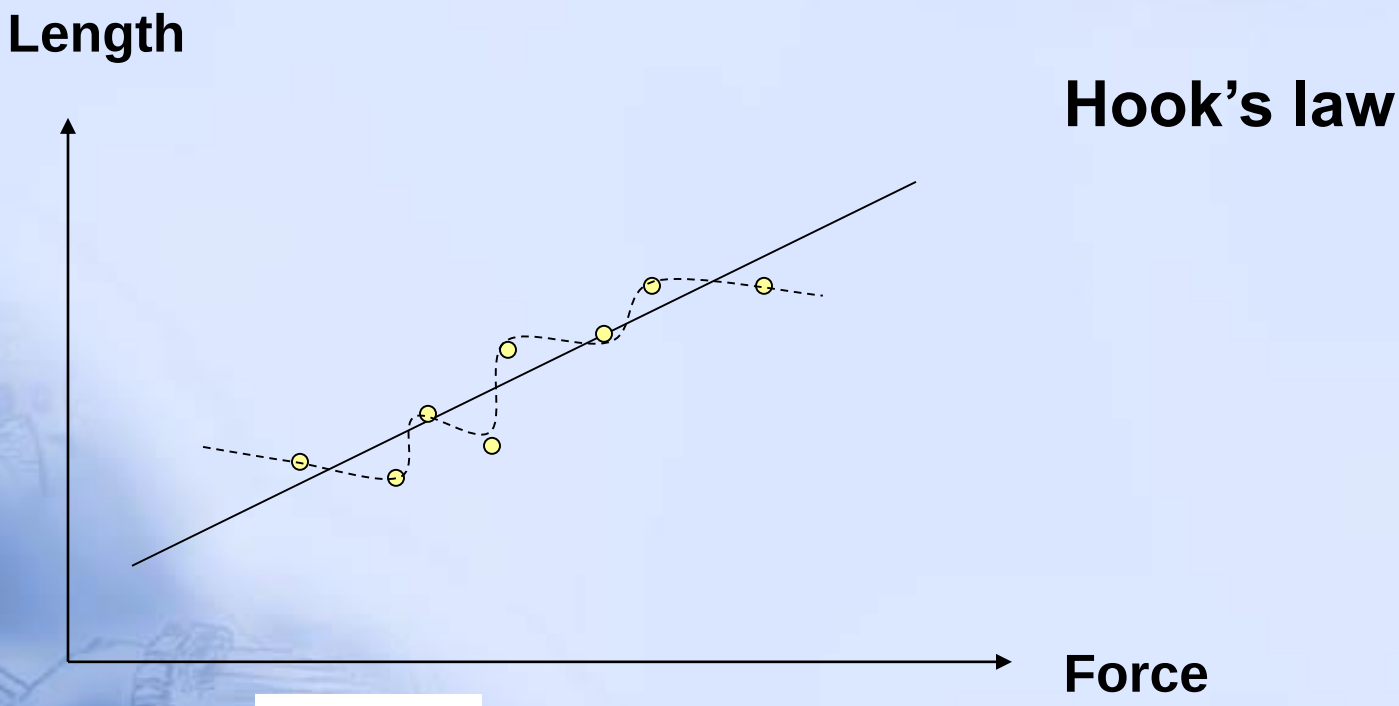
- SVM: hinge loss minimization + regularization

Boosting

- Boosting: (implicit) large l_1 / l_∞ margin classifier
- Boosting: exp loss minimization (+ regularization)

- “Classical” Machine Learning Theories
  - VC theory — Capacity of the hypothesis space
  - PAC-theory
  - Margin theory — Confidence
  - Empirical Processes — Capacity
  - PAC-Bayes theory — PAC in Bayes framework
  - Regularization — Capacity, smoothness

ML theories: Quantification of Occam's Razor



- Comparison of “Classical” Machine Learning Theories
  - Regularization:
    - Bayesian optimality
    - Only asymptotic (convergence, rate, non-uniform)
  - VC/PAC, Margin, PAC-Bayes,…
    - Relative optimality (optimal in a hypothesis space)
    - Non-asymptotic (finite sample bounds)

- Limitations of the “Classical” ML
  - Representation
    - Euclidean representation for input.
    - Simple representation for output.
- How to represent STRUCTURES in data?

Outline

- A brief overview of Machine Learning
- Graphical Models
  - Representation
  - Inference
  - Learning

Chapter I: Representation

Probabilistic Graphical Models: What and Why

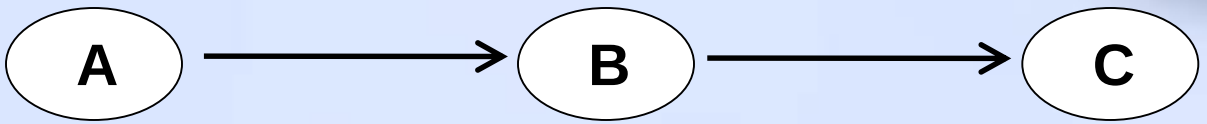
- PGMs:
  - A model for joint probability distribution over random variables.
  - Represent dependencies and independencies between the random variables.
- Why is probability distribution important?
  - Genes and diseases, and everything
- Why PGM was invented by computer scientist, why not the statisticians?

- **Two types of PGMs**
  - **Directed graph: Bayesian Networks (BNs).**
  - **Undirected graph: Markov Random Fields (MRFs)**

Bayesian Networks (BNs)

- **(Intuitively) How BNs Represent Joint pdfs:**

Example 1:



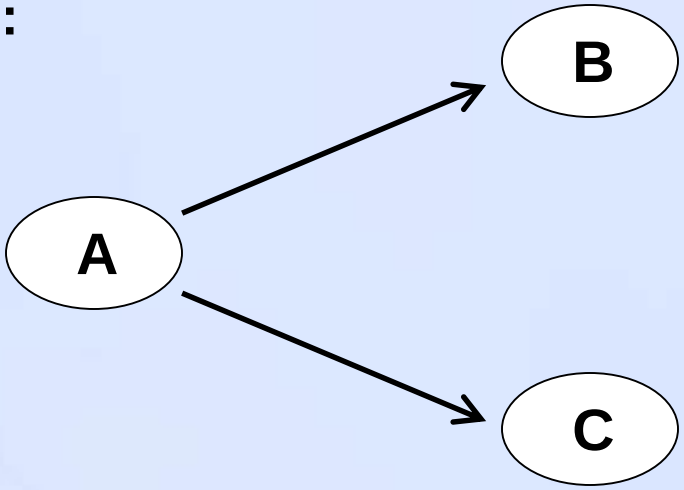
$$P(ABC) = P(A)P(B \mid A)P(C \mid B)$$

Given B, C and A are independent

Note: Dependency vs. Causality

- **(Intuitively) How BNs Represent Joint pdfs:**

Example 2:

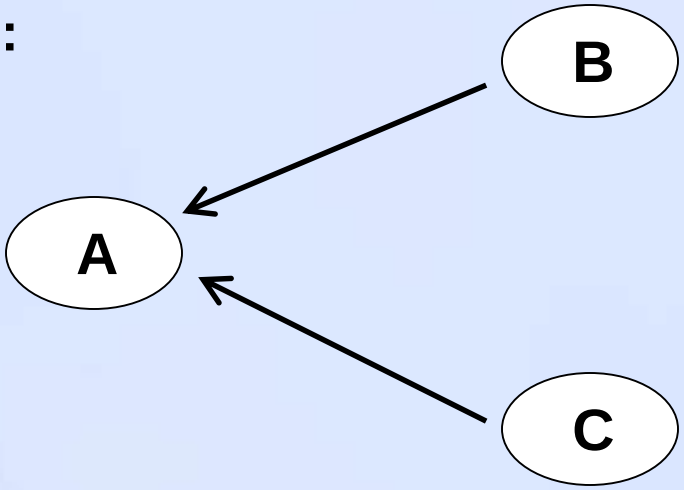


$$P(ABC) = P(A)P(B \mid A)P(C \mid A)$$

Given A, B and C are independent

- **(Intuitively) How BNs Represent Joint pdfs:**

Example 3:



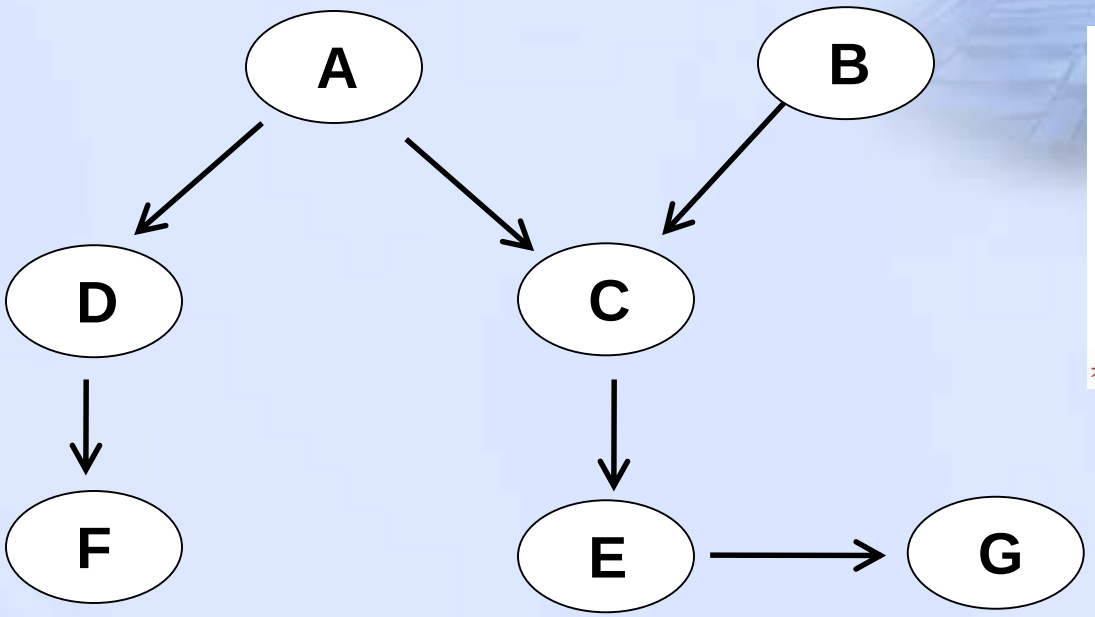
$$P(ABC) = P(B)P(C)P(A|BC)$$

B and C are independent;

But given A, B and C are NOT independent

- **(Intuitively) How BNs Represent Joint pdfs:**

Example 4:

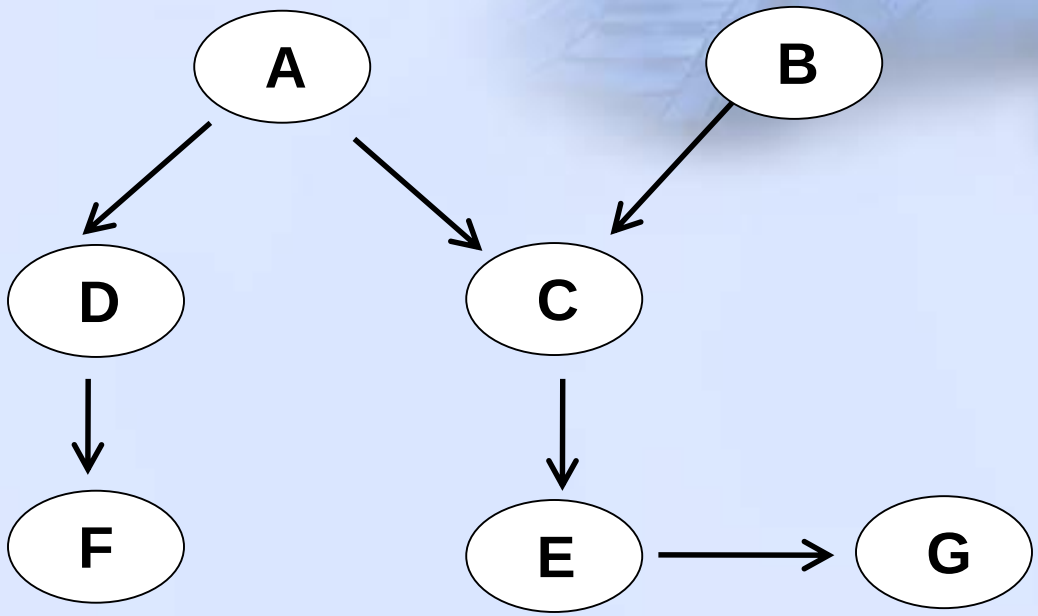


$$P(ABCDEFG)$$
$$= P(A)P(B)P(C \mid AB)P(D \mid A)P(E \mid C)P(F \mid D)P(G \mid E)$$

$$P(ABCDEFG)$$
$$= P(A)P(B)P(C \mid AB)P(D \mid A)P(E \mid C)P(F \mid D)P(G \mid E)$$

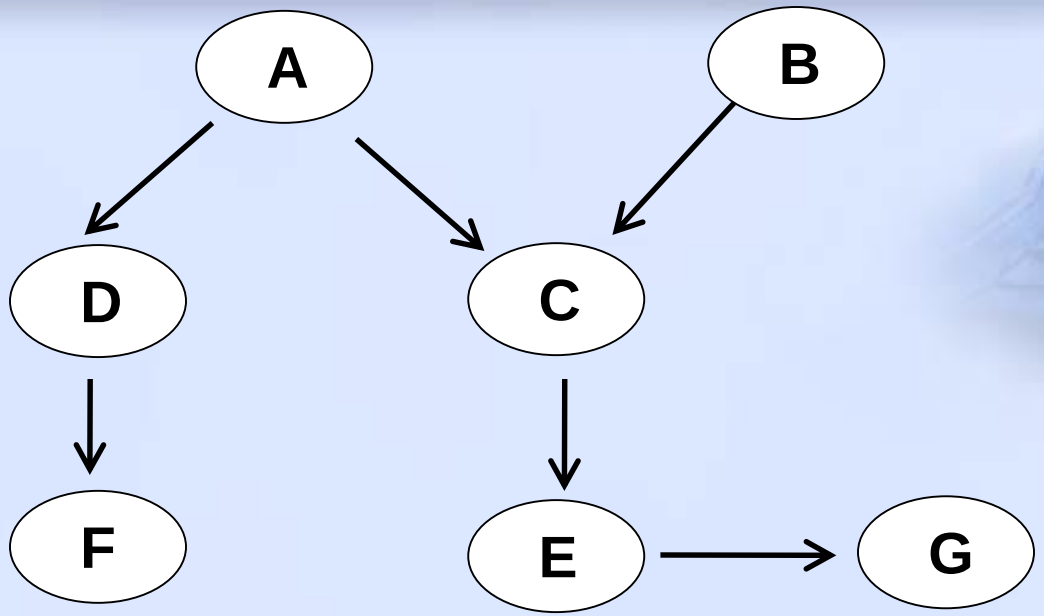
- **Learning:**

  Find a factorization rule according to previous examples.



- **Factorization:**

$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$



$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

The graph must be **acyclic!**

BN must be **DAG**

- Definition (**Factorize according to a DAG**):

A probability distribution P is said to be factorized according to a directed acyclic graph G if

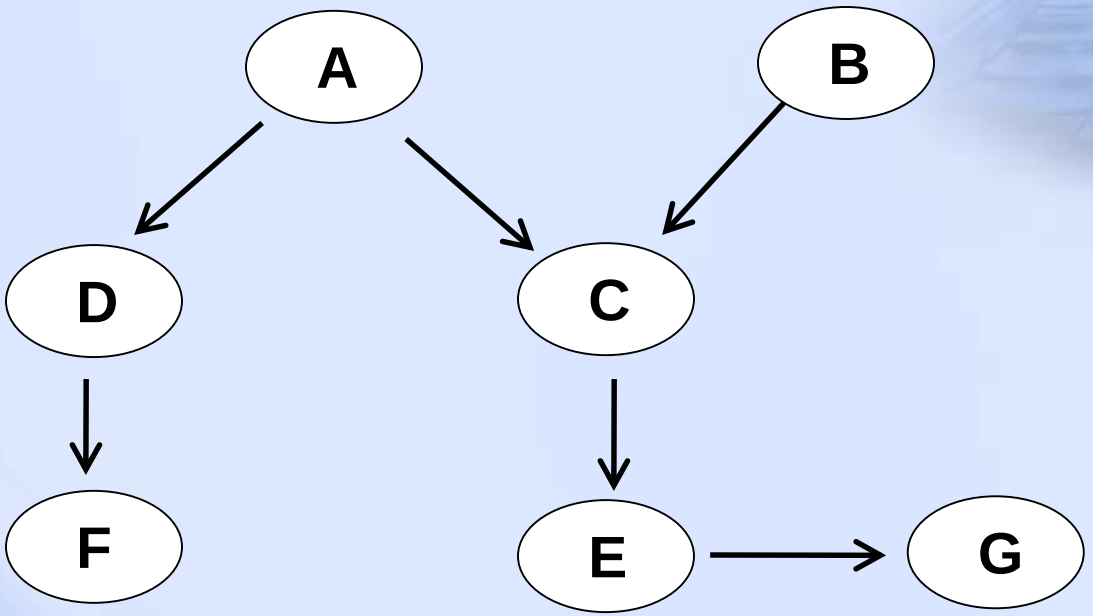
$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

- Definition (**Bayesian Network**):

A Bayesian network is a pair (P, G) of a probability distribution and a DAG, where P is factorized according to G, and all the “local” conditional probability distributions are given.

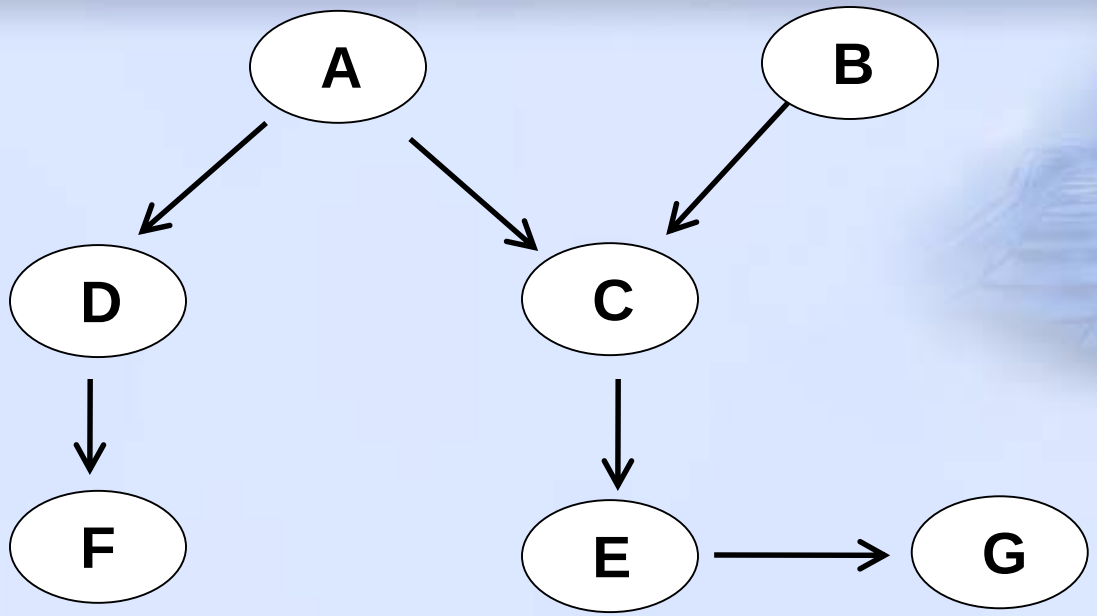
$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

- **Given the factorization, which variables are independent of C, given C's Parents A and B?**



$$P(ABCDEFG)$$
$$= P(A)P(B)P(C \mid AB)P(D \mid A)P(E \mid C)P(F \mid D)P(G \mid E)$$

$$D, \quad F$$



$$P(ABCDEFG) = P(A)P(B)P(C|AB)P(D|A)P(E|C)P(F|D)P(G|E)$$

$$P(CD|AB) = P(C|AB)P(D|AB)$$

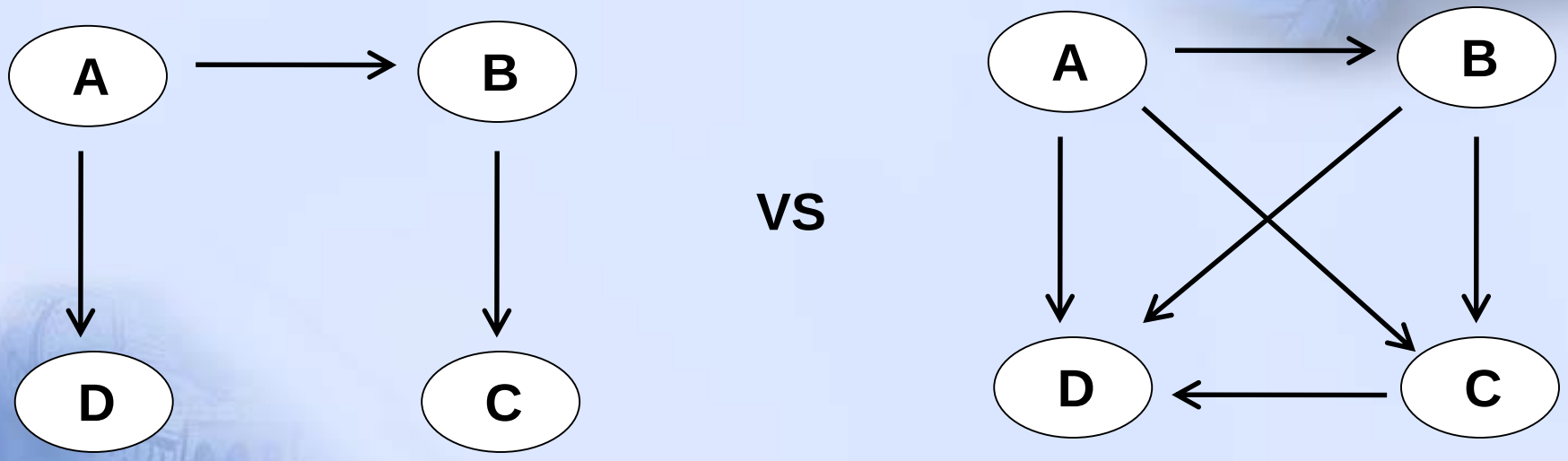
$$P(CF|AB) = P(C|AB)P(F|AB)$$

- Question: Let C be a node (random variable) in a BN. Which nodes (random variables) are independent of C, given C's Parents?

- Theorem (**Local Markov property for BN**):

  For any node C (random variable) in a BN, all nodes that are not descendents of C are independent of C, given C's parents.

Sparse vs. Dense



- What is the joint pdf of the right BN?
- Is there any independence in the right BN?

- Question: Given a BN=(P, G), can you determine, for three arbitrary sets of random variables X={…}, Y={…}, and Z={…}, whether the following conditional independency hold?

$$X \perp Y \mid Z$$

- Definition (**active trail in BN**)

  Let x, y be two nodes and Z be a set of nodes. A path between x and y are said to be an active trial given Z, if the followings are true:

1) Let $x \Leftrightarrow \ldots \Leftrightarrow x' \rightarrow m \leftarrow y' \Leftrightarrow \ldots \Leftrightarrow y$ be the path, then m or one of its descendants is not in Z. That is, whenever there is a "**v-structure**" in the path, the middle node or one of its descendants is in Z;

2) No other node along the path is in Z.

- Definition (**D-separation in BN**)

  Let X, Y, Z be three sets of nodes. X and Y are said to be D-separated by Z if for every node x in X and every y in Y, and every path between x and y, the path is not an active trial given Z.

- Theorem (Informal)

  The independencies in a BN are exactly those characterized by D-separation.

- Theorem

For any BN (P,G), and arbitrary sets of nodes X, Y, Z. If X and Y are D-separated by Z in G, then

$$P(X,Y \mid Z) = P(X \mid Z)\ P(Y \mid Z)$$

- Theorem

For any DAG G, and any sets of nodes X, Y, Z. If X and Y are not D-separated by Z in G, then there must exist a probability distribution P which factorize according to G, but

$$P(X,Y \mid Z) \neq P(X \mid Z)\ P(Y \mid Z)$$

The Representation Limit of BN

- **Is there a BN that has precisely these independencies?**

$$A \perp B | C, D; \qquad C \perp D | A, B;$$

- **Not every distribution can be represented by a BN satisfying exactly all the independencies!**

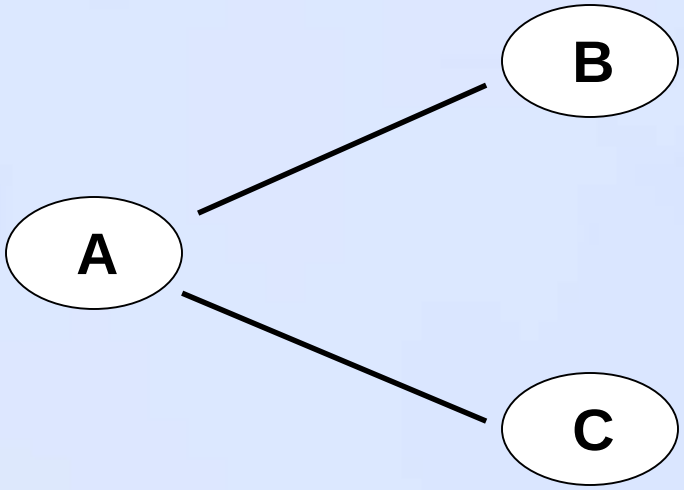
- To sum up:
  - BN represents joint probability distributions that can factorize according to:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

  - The local independencies in BN are characterized by parents and non-descendants.
  - The global independencies in BN are characterized by D-separation.

Markov Random Fields (MRF)

- **How MRFs Represent Joint pdfs:**

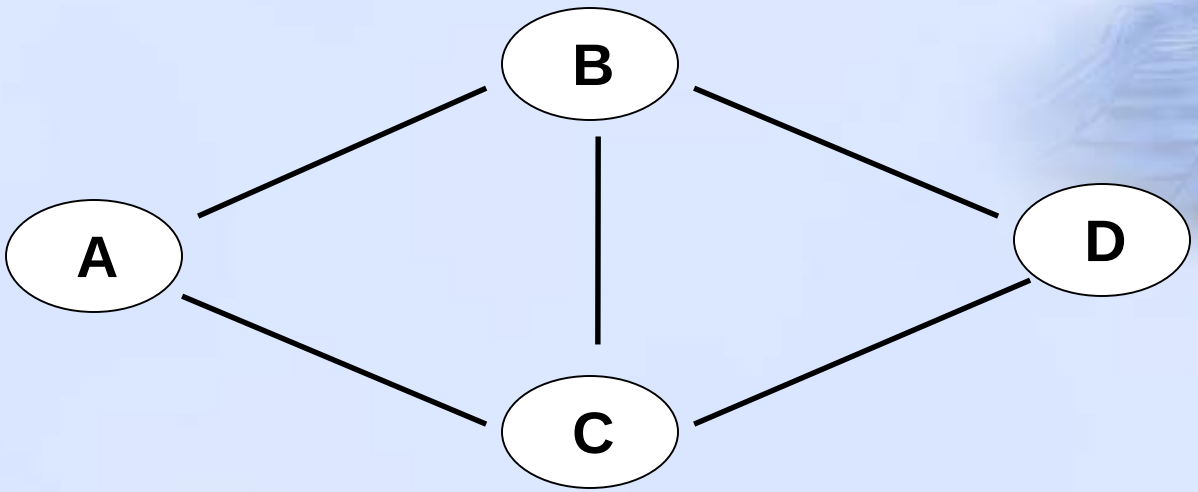


factors

$$P(ABC) = \frac{1}{Z}\phi_1(AB)\phi_2(AC).$$

Partition function

- **How MRFs Represent Joint pdfs:**



$$P(ABCD) = \frac{1}{Z}\phi_1(ABC)\phi_2(BCD).$$

- **Factors correspond to *maximal* cliques.**
- **The joint distribution is the product of all factors normalized by the partition function.**

- Formal Definition (**MRFs**):

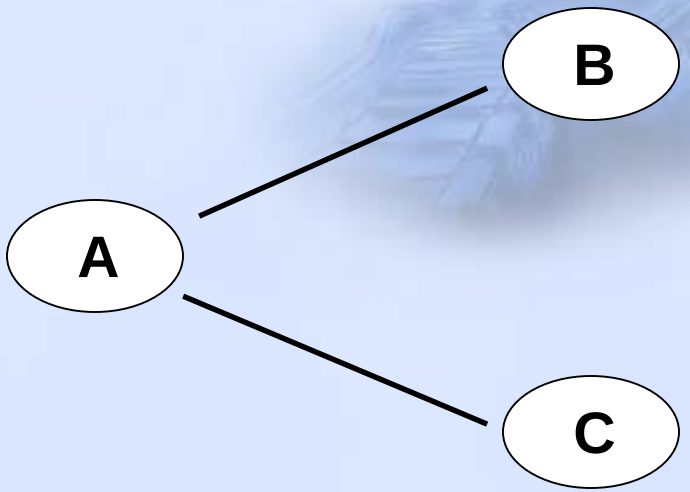
  A Markov network is a pair (P, G), where G is an undirected graph and P factorizes according to G, i.e., P has the form

$$P(X_1,..., X_n) = \frac{1}{Z}\tilde{P}(X_1,..., X_n) = \frac{1}{Z}\prod_i \phi_i(C_i)$$

  where each C_i is a (maximal) clique in G.

- **The independence in MRF:**

$$P(ABC) = \frac{1}{Z}\phi_1(AB)\phi_2(AC).$$



Easy to see:

$$B \perp C|A$$

$$P(BC|A) = P(B|A)P(C|A)$$

- Question: Given a MRF=(P, G), can you determine, for three arbitrary sets of random variables X={…}, Y={…}, and Z={…}, whether the following conditional independency hold?

$$X \perp Y \mid Z$$

- Definition (Separation)

  Let X, Y, Z be three sets of nodes in an undirected graph G. X and Y are said to be separated by Z if for every node x in X and every y in Y, and every path between x and y, there is a node in the path that belongs to Z.

- Theorem (Informal)

  All independencies in MRF are characterized by separation.

- Theorem

For any MRF (P,G), and arbitrary sets of nodes X, Y, Z. If X and Y are separated by Z in G, then

$$P(X,Y \mid Z) = P(X \mid Z)\ P(Y \mid Z)$$

- Theorem

For any undirected graph G, and any sets of nodes X, Y, Z. If X and Y are not separated by Z in G, then there must exist a probability distribution P which factorize according to G, but

$$P(X,Y \mid Z) \neq P(X \mid Z)\ P(Y \mid Z)$$

Machine Learning and Graphical Models

(Lecture II)

王立威

北京大学

信息科学技术学院

http://www.cis.pku.edu.cn/faculty/vision/wangliwei/

wanglw@cis.pku.edu.cn

Outline

- A brief overview of Machine Learning
- Graphical Models
  - Representation
  - Learning
  - Inference

Chapter II: Learning

- Learning Graphical Models:

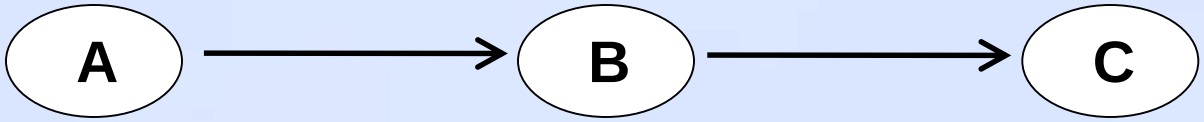
  - Definition of **Bayesian Networks**:

A Bayesian network is a pair (P, G) of a probability distribution and a DAG, where P is factorized according to G, and all the “local” conditional probability distributions are given.

$$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

- Learning Bayesian Networks
  - BN = **Structure** (graph) + **Local conditional distribution**
  - Learning BN:
    - How to learn distribution's **parameters** from data
      - Relatively simple, standard parameter estimation.
    - How to learn **structure** from data
      - Very difficult ! Why?

- Structure learning:
  - Structure (graph): **edges** between nodes.
  - Is the structure of a BN learnable?



  - Note: the edge A —> C exists or not equals to whether the following equation holds strictly.

$$P(AC \mid B) = P(A \mid B)P(C \mid B)$$

- Useful structure learning methods:
  - Constraint-based structure learning.
    - Using hypothesis test to obtain independencies.
    - Construct the graph.
  - Score-based structure learning: Penalizing “dense” graph
    - Likelihood scores
    - BIC
    - MDL
    - AIC

Open Problems

- Robust Learning of Structures?

Outline

- A brief overview of Machine Learning
- Graphical Models
  - Representation
  - Learning
  - Inference

Chapter II: Inference

- What is inference in GM
- The hardness of inference in GM
- Exact inference algorithms
- Approximate inference algorithms
- Future research directions

What is inference in GM

- Input: a graph and the local conditional distributions.

- Goal: two types of inference
  - Conditional probability

$$\Pr(X = x \mid E = e)$$

  - MAP inference

$$\max_x \Pr(X = x \mid E = e)$$

The hardness of inference in GM

- Exact inference in GM is hard
  - Decision version of exact inference is **NP**C.
  - Exact inference is #**P** complete.

- Approximate inference in GM is hard
  - ε-approximate inference is **NP**-hard for every ε

- Thm.1: Decide $\Pr(X = x) > 0$ is **NP** complete.

  Proof: Reduction from 3SAT.

- Thm.2: Compute $\Pr(X = x)$ is **#P** complete.

  Proof: Use above reduction from #3SAT. A Levin reduction, certificates are one-to-one.

- Thm.3: For every $\varepsilon > 0$, compute an ε-approximate of $\Pr(X = x)$ is **NP**-hard.

Proof: $\hat{p}$ is an ε-approximation of $\Pr(X = x)$ means that

$$\frac{\Pr(X = x)}{1+\varepsilon} \leq \hat{p} \leq \Pr(X = x)(1+\varepsilon)$$

Clearly, if one has an ε-approximation of $\Pr(X = x)$, one can solve the **NP**C problem $\Pr(X = x) > 0$.

Remark: For absolute approximate it's still **NP**-hard.

- Thm.4: Exact and approximate MAP inference are all hard.

- **Conclusion**: The worst-case complexity of the inferences, both exact and approximate are **NP-**hard.

Exact Inference Algorithms

- The relation between BN and MRF
  - From BN to MRF (factors)

BN: $$P(X_1, \ldots, X_n) = \prod_i P(X_i | Pa(X_i))$$

MRF: $$P(X_1,\ldots, X_n) = \frac{1}{Z}\tilde{P}(X_1,\ldots, X_n) = \frac{1}{Z}\prod_i \phi_i(C_i)$$

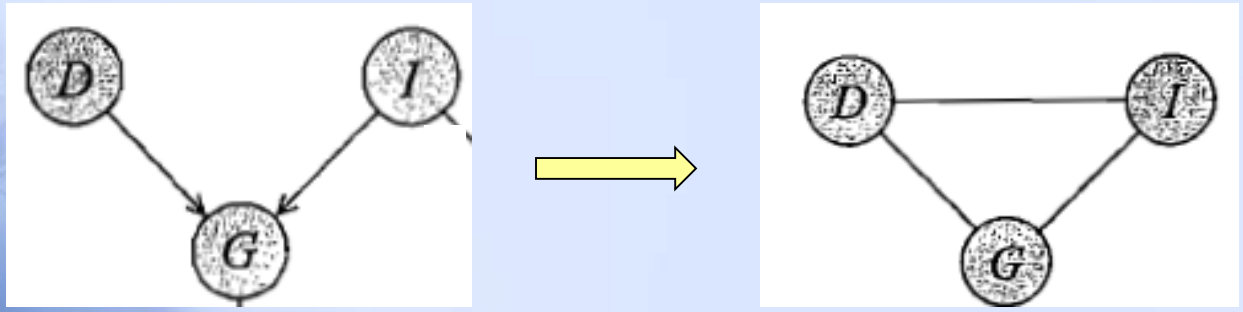
BN --> MRF: $\phi_i(X_i, Pa(X_i)) = P(X_i \mid Pa(X_i))$

$$Z = 1$$

- From BN to MRF (graphs)
  - Moral graph （联姻图）

    Delete the directions for edges;

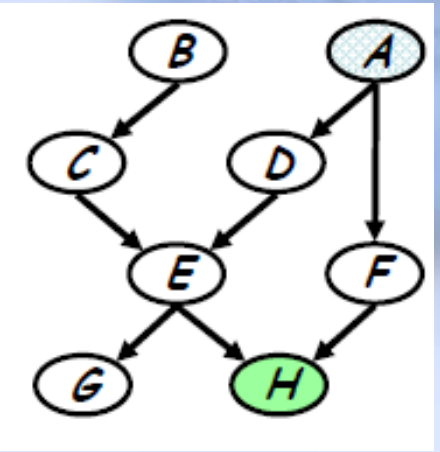
    Connecting the parents.



- Exact inference algorithms:
  - The variable elimination algorithm.
  - Belief propagation: the sum-product algorithm.
  - Belief update.

- Why study exact inference algorithms?
  - Gain intuition and insight for developing useful approximate algorithms.

- The variable elimination algorithm



query: $\Pr(A = a \mid H = h)$

distribution:

$$P(A)P(B)P(C \mid B)P(D \mid A)P(E \mid C, D)P(F \mid A)P(H \mid E, F)P(G \mid E)$$

solve:

$$P(a, h)$$

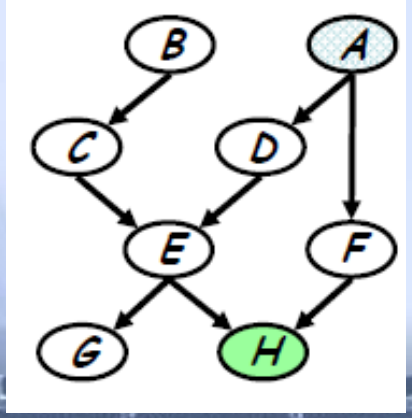
$$= \sum_{b,c,d,e,f,g} P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(h \mid e, f)P(g \mid e)$$

Variable elimination---dynamic programming

$$P(a,h) = \sum_{b,c,d,e,f,g} P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(h|e,f)P(g|e)$$

$$= P(a)\sum_{b} P(b)\sum_{c} P(c|b)\sum_{d} P(d|a)\sum_{e} P(e|c,d)\sum_{f} P(f|a)\sum_{g} P(g|e)P(h|e,f)$$

$$\phi(a) \quad \phi(b) \quad \phi(c,b) \quad \phi(d,a) \quad \phi(e,c,d) \quad \phi(f,a) \quad \phi(g,e) \quad \phi(h,e,f)$$



$$\phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\sum_d \phi(d,a)\sum_e \phi(e,c,d)\sum_f \phi(f,a)\phi(h,e,f)\sum_g \phi(g,e)$$

Step 1: $\delta(e) = \sum_g \phi(g,e) = 1$ **(g,e)**

$$\Rightarrow \phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\sum_d \phi(d,a)\sum_e \phi(e,c,d)\sum_f \phi(f,a)\phi(h,e,f)(\delta(e))$$

Step 2: $\delta(a,e) = \sum_f \phi(f,a)\phi(h,e,f)(\delta(e))$ **(a,e,f)**

$$\Rightarrow \phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\sum_d \phi(d,a)\sum_e \phi(e,c,d)\delta(a,e)$$

$$\phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\sum_d \phi(d,a)\sum_e \phi(e,c,d)\delta(a,e)$$

Step 3: $\delta(a,c,d) = \sum_e \phi(e,c,d)\delta(a,e)$ **(a,c,d,e)**

$$\phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\sum_d \phi(d,a)\delta(a,c,d)$$

Step 4: $\delta(a,c) = \sum_d \phi(d,a)\delta(a,c,d)$ **(a,c,d)**

$$\phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\delta(a,c)$$

$$\phi(a)\sum_b \phi(b)\sum_c \phi(c,b)\delta(a,c)$$

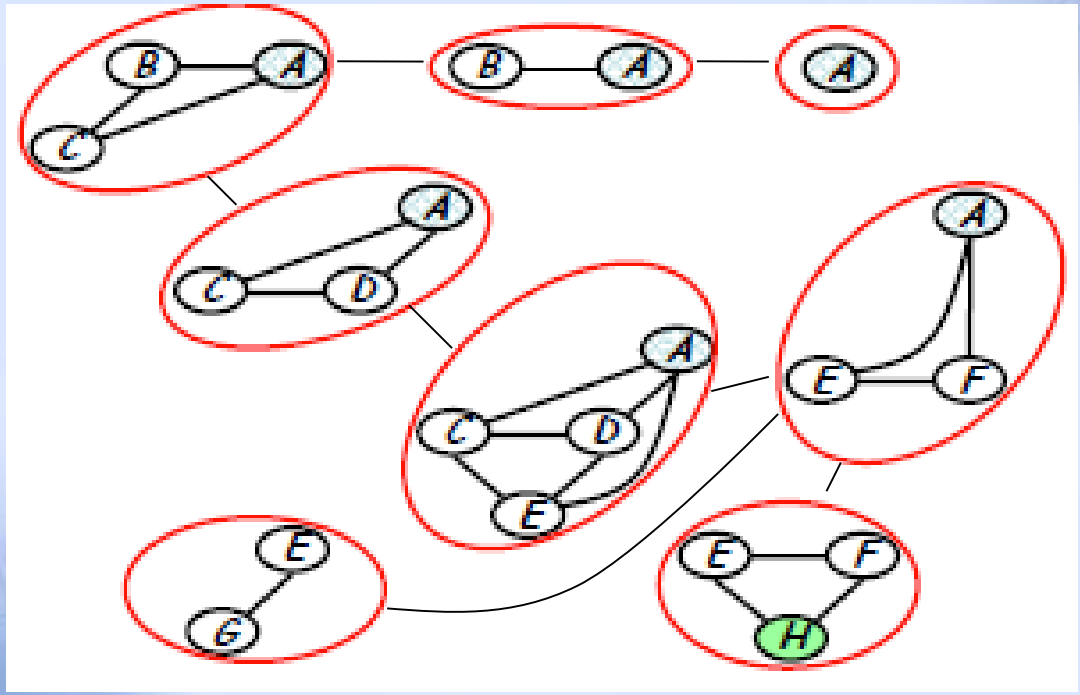
Step 5: $\delta(a,b) = \sum_c \phi(c,b)\delta(a,c)$ **(a,b,c)**

$$\phi(a)\sum_b \phi(b)\delta(a,b)$$

Step 6: $\delta(a) = \sum_b \phi(b)\delta(a,b)$ **(a,b)**

$$\phi(a)\delta(a)$$

- Message Passing: the SP algorithm
  - Variable elimination induced clique tree



Family preserving property: factors scope

Running intersection property!

Variable elimination as message passing on clique tree:

- General message passing from clique C_i to C_j

$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

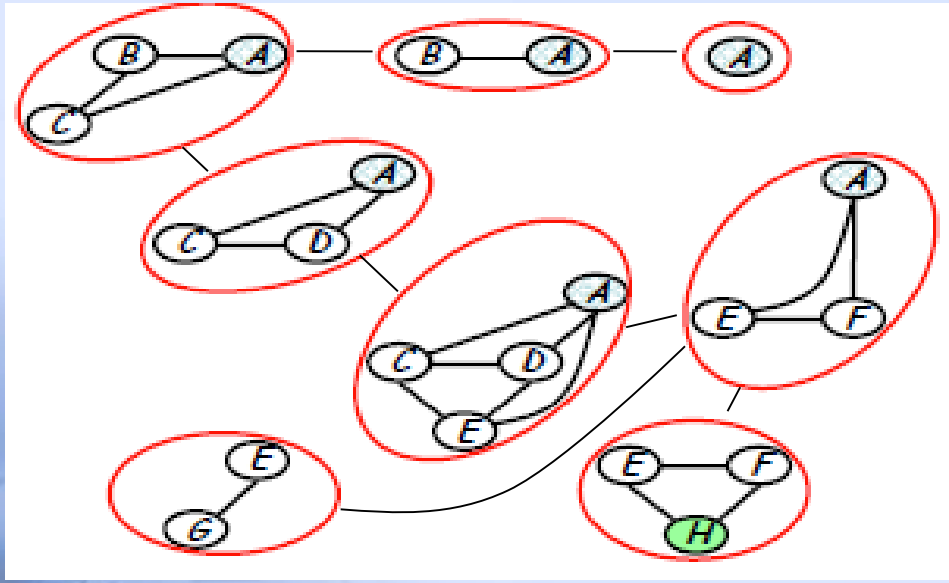
$$S_{ij} = C_i \cap C_j$$

ψ_i : original terms (potential) of C_i

- The Sum-Product belief propagation algorithm:
  - Construct a clique tree that satisfies the family preserving and **running intersection properties**.
  - Choose clique where the query variable lies in as the root.
  - Message passing from the leaves.
  - After all messages arrive at the root, sum over all other variables in the root clique other than the query variable.

Thm: The Sum-Product algorithm always gives the correct answer for any clique tree!

- What if we want to compute k queries?
  - Message passing k times?
  - No! Message can be reused.
  - For each **edge** in the clique tree, twice is enough, one for each direction (up-down)



- After message passing on all edges in two directions, each clique has a belief (joint probability).

- Belief: $\beta_i = \psi_i \prod_{k \in Nb(i)} \delta_{k \to i}$

- The system must satisfy the **calibration** property:

$$\sum_{C_i \setminus S_{ij}} \beta_i = \sum_{C_j \setminus S_{ij}} \beta_j$$

- Why calibration holds?

$$\beta_i = \tilde{P}(C_i) \propto P(C_i)$$

Beliefs are marginal probabilities on the cliques!

$$\sum_{C_i \setminus S_{ij}} \beta_i = \sum_{C_j \setminus S_{ij}} \beta_j = \tilde{P}(S_{ij}) \propto P(S_{ij})$$

Agree on the marginal probability of S_{ij} !

- Belief update algorithm:
  - Message passing:

$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

  - Belief: $\beta_i = \psi_i \prod_{k\in Nb(i)} \delta_{k\to i}$

  - So $\delta_{i\to j} = \dfrac{\sum_{C_i \setminus S_{ij}} \beta_i}{\delta_{j\to i}}$

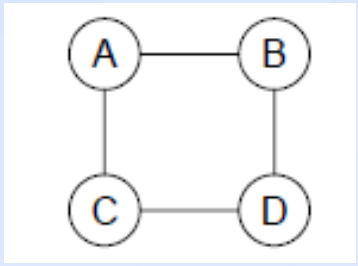
    Propagating beliefs from C_i to C_j

- Belief update algorithm:
  - Construct clique tree
  - Initialize: $\beta_i = \psi_i \qquad \mu_{i \to j} = 1$
  - Update: $t_{i \to j} \leftarrow \sum_{C_i \setminus S_{ij}} \beta_i$

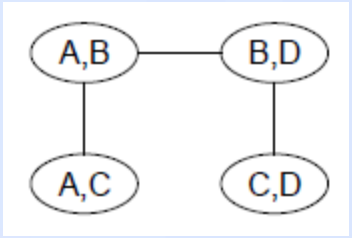
$$\beta_j \leftarrow \frac{\beta_j t_{i \to j}}{\mu_{ij}}$$

$$\mu_{ij} \leftarrow t_{i \to j}$$

- How to construct clique tree?
  - Graph:

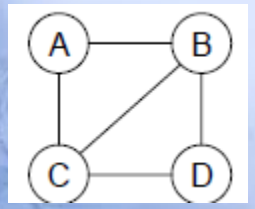


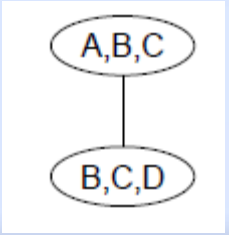
  - Clique tree?



No!

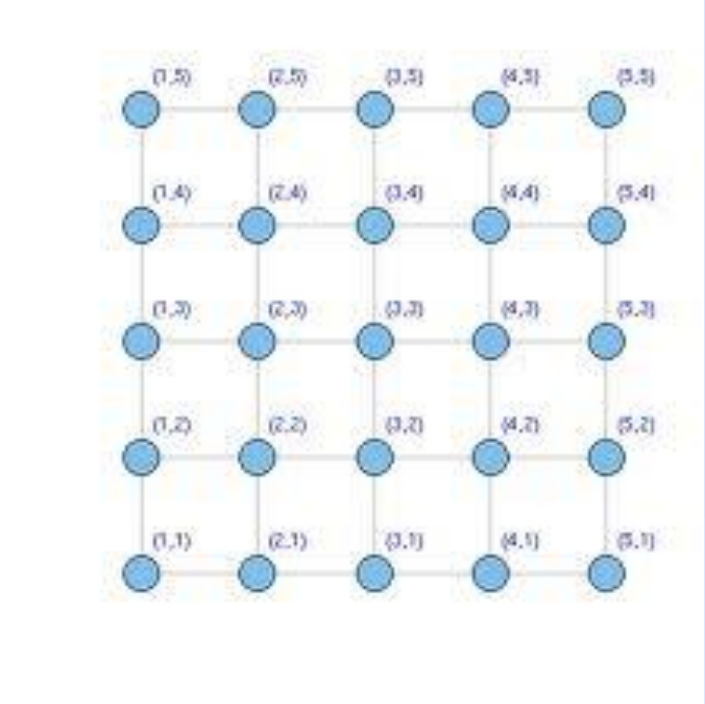
  - The graph cannot have a 4-loop (or larger)!
  - Solution: triangulation and chordal graph.





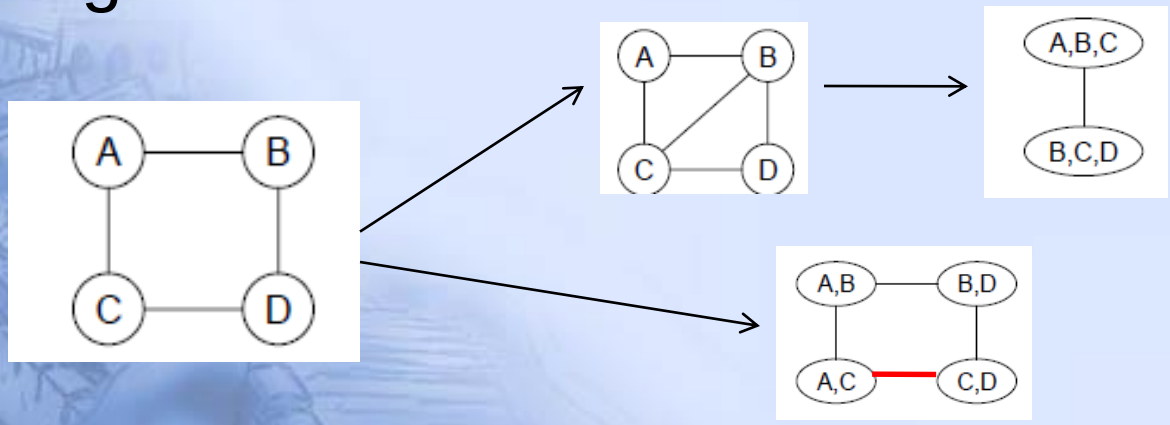
- What if the graph is an Ising model,

  Triangulation and chordal graph induce clique trees that have large width and the computational complexity is very high.



- Construction of Clique Trees:
  - Variable Elimination;
  - BN to Moral graph, then to chordal graph, then find all the maximum cliques, and then use the maximum spanning tree algorithm to construct the edges of the clique tree.

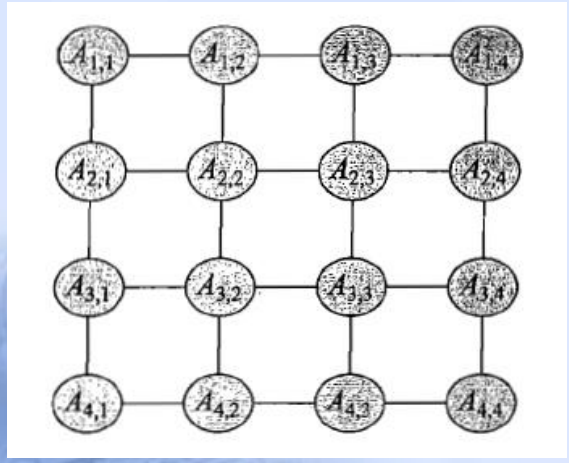
- Reflection of the Sum-Product Belief Propagation algorithm:
  - The algorithm itself does require the clique tree has to be a “tree”.
  - What if we construct a general “clique graph” instead of a “clique tree” and run the SP algorithm?

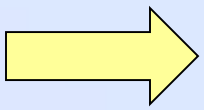


Clique tree

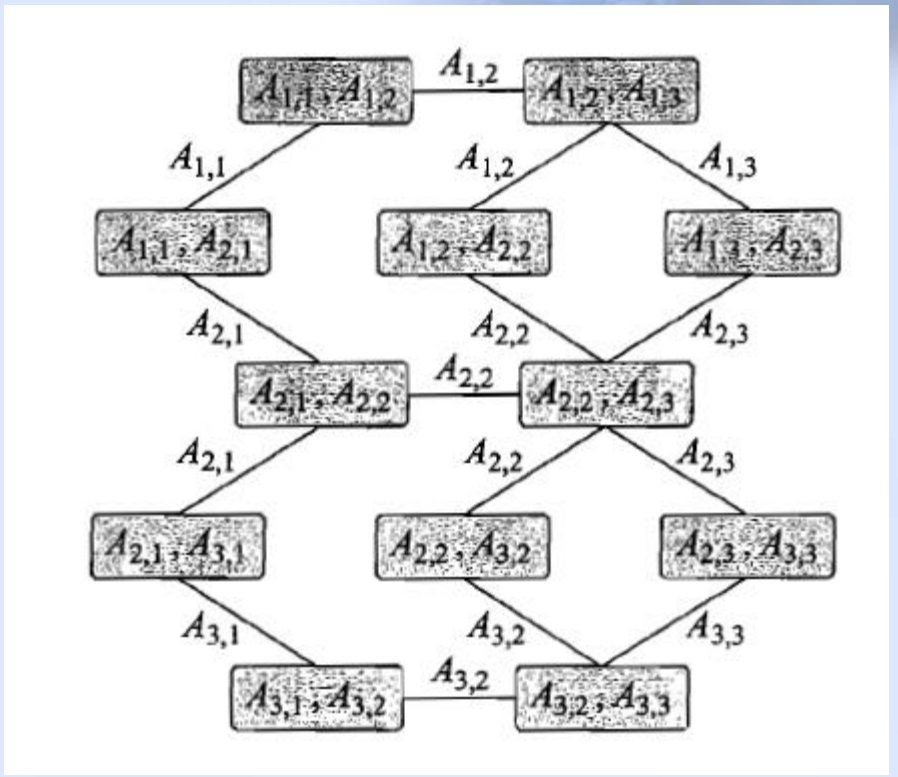
Cluster graph

Ising Model





Cluster Graph



- What's the result if run Sum-Product belief propagation on the cluster graph?

  Loops?

  Convergence?

  Correctness?

  This is the "Loopy Belief Propagation" algorithm!

Machine Learning and Graphical Models

(Lecture III)

王立威

北京大学

信息科学技术学院

http://www.cis.pku.edu.cn/faculty/vision/wangliwei/

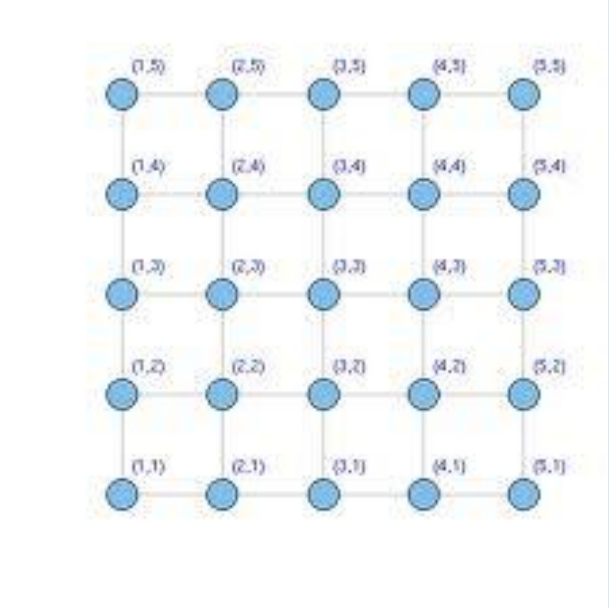
wanglw@cis.pku.edu.cn

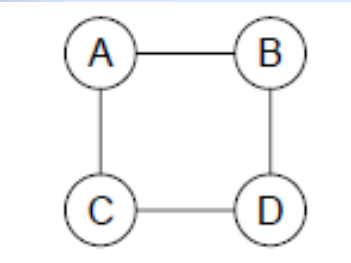
- What is inference in GM
- The hardness of inference in GM
- Exact inference algorithms
- Approximate inference algorithms
- Future research directions

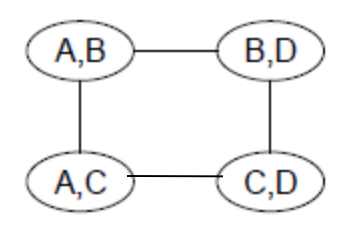
- Approximate inference methods:
  - Variational Methods.
    - (Loopy) belief propagation.
    - Mean field method.
  - Sampling based methods.
    - Metropolis-Hastings
    - Gibbs Sampling
    - Importance Sampling

Variational Methods

- (Loopy) belief propagation:







Cluster graph

- Loopy belief propagation: just let the belief propagates.
- On clique trees, belief propagation converges after propagating on all edges (two directions).
- For general cluster graphs, it is not guaranteed to converge. Even if it converges, it can converge to a wrong answer.

- Variational explanation of the (loopy) belief propagation:
  - Exact inference

    Proposition: The following optimization problem gives the same result as exact inference:

    Find a set of beliefs to minimize the Helmholtz free energy (or equivalently the relative entropy) to the factored distribution on clique trees under the constraints of calibration (local agreement).

- The fixed point characterization of the solution of the optimization problem:

$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

- Approximate inference

  Find a set of beliefs to minimize the factored free energy to the factored distribution on cluster graphs under the constraints of calibration.

- The fixed point characterization of the solution of this optimization problem is the (loopy) belief propagation formula.

- Mean field method:
  - Using first order approximation of the target distribution:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i)$$

  - The fixed point equation and updating rule have simple forms.

Sampling Based Methods

- Sampling-based (Monte Carlo) algorithms:
  - Idea: Using sample based frequency to approximate the probability.
  - Probability is Expectation:

$$P(X = x) = E[I(X = x)]$$

  - Expectation (of a function) can be approximated by sample average:

$$E[f(X)] \approx \frac{1}{n}\sum_i f(X_i)$$

$$X, X_1, \cdots, X_n \quad i.i.d. \sim P$$

- Monte Carlo is useful in many problems:
  - High dimensional numerical integration.
- How to generate the sample when the target probability distribution is difficult to compute?
- Markov Chain Monte Carlo (MCMC)
  - Key idea :Generate data by Markov chain, whose stationary distribution is the target pdf.

- A brief review of finite state Markov chains:
  - Reducible vs. Irreducible
  - Periodic vs. Aperiodic
  - **Ergodic**: no matter which initial state is, the process will converge to a unique stationary distribution.

- **Regular MC**: 1) for every pair of states x, x', the prob. that x will reach x' in k steps for some finite k is positive; 2) for every state x, there is positive prob. that x will stay in x in the next step.

  Easy to understand via the graph view

- **Theorem** (sufficient condition for ergodic MC):

  If a finite state MC is regular, then it is ergodic.

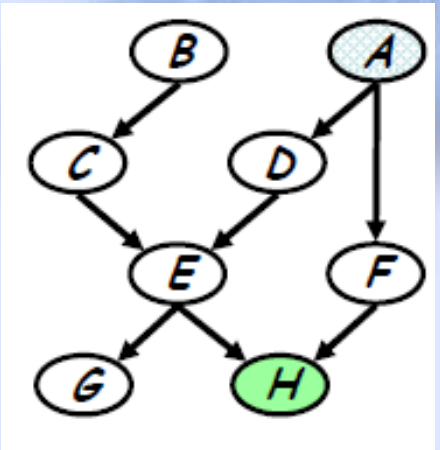
- Regular Markov chains are good for our purpose.
- We run the Markov chain, and wait for it converges to the stationary distribution, then the data can be used for approximate calculation.
- But, how long will it take for the MC to converge?

Gibbs Sampling

- Gibbs Sampling
  - One of the simplest sampling algorithm.
  - Assume for $X = (X^1, \cdots, X^d)$, $P(X^j \mid X^{-j})$ is easy to sample.
  - Easy to implement for Graphical Models.
  - Proposed by Geman & Geman (TPAMI, 1984).

- Gibbs sampling algorithm
  - Goal: draw a sequence of examples $x_1, x_2 \cdots, x_n$, when $n \to \infty$, $x_n \sim P$, where P is the target distribution; $x_i = (x_i^1, \cdots, x_i^d) \in R^d$
  - Algorithm:
    - Draw from some initial distribution $x_0 \sim P_0$
    - For $t = 1, \cdots, n$
      - 1. $x_t \leftarrow x_{t-1}$
      - 2. For each $j \in [d]$
        - Sample x_t^j according to $P(X^j \mid X^{-j})$
    - Return $x_1, x_2 \cdots, x_n$

- Why Gibbs sampling is easy for PGM?
  - Sample $P(F \mid a,...e,g,h)$



$$P(F \mid a,...e,g,h)$$

$$= \frac{P(F,a,...e,g,h)}{\sum_f P(f,a,...e,g,h)}$$

$$= \frac{P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(F \mid a)P(g \mid e)P(h \mid e,F)}{\sum_f P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)}$$

$$= \frac{P(F \mid a)P(h \mid e,F)}{\sum_f P(f \mid a)P(h \mid e,f)}$$

Only the factors that involve F left!

- Generally, for both Bayesian networks and Markov networks, the conditional probability for Gibbs sampling involves only factors that the query random variable lives in.

- Trivially generalize to the case where there is evidence
  - Draw a sequence of examples where the target distribution is $P(\cdot \mid E = e)$

- **Theorem**:

  The Gibbs sampling process has a unique stationary distribution P (or $P(\cdot \mid E = e)$)

- Disadvantages of Gibbs sampling for PGMs:
  - Slow convergence to stationary distribution.

Metropolis-Hastings Algorithm

- For Gibbs sampling, we assume that it is easy to generate a sample from $P(X^j \mid X^{-j})$. But sometimes, this is difficult.

- More generally, for a target distribution P , it may be very difficult to generate sample directly according to P , does MCMC help?

- The idea of Metropolis-Hastings:
  - Using a *proposal* distribution $\tilde{T}(x' \mid x)$: a transition model.

- An important result for Markov chain:
  - Detailed Balance (Reversible Markov chain):

    Definition: A finite state MC with transition probability matrix T is said to be *reversible* if there exists a unique distribution P such that for all x, x'

$$P(x)T(x'|\,x) = P(x')T(x\,|\,x').$$

    The above equation is called ***detailed balance***.

  - Reversible: for any sequence $x_1, x_2 \cdots, x_n$, the probability that it occurs in the process is the same as the probability that $x_n, x_{n-1} \cdots, x_1$ occurs.

- **Theorem**:

  If the transition matrix T defines a regular Markov chain, and T satisfies the detailed balance w.r.p. to P, then P is the unique stationary distribution of the Markov chain T.

- Metropolis-Hastings algorithm:
  - Goal: draw a sequence of examples $x_1, x_2 \cdots, x_n$, when $n \to \infty$, $x_n \sim P$, where P is the target distribution.
  - Algorithm:
    - Let $\tilde{T}(x' \mid x)$ be a proposal transition model.
    - Define the transition matrix of a Markov chain as:

$$T(x' \mid x) = \min\left[1, \frac{P(x')\tilde{T}(x \mid x')}{P(x)\tilde{T}(x' \mid x)}\right]$$

    - Generate $x_1, x_2 \cdots, x_n$ according to the MC of T.

- Proposition:

  For any target distribution P, and any proposal transition model $\tilde{T}(x'|x)$, the Markov chain defined by T in the Metropolis-Hastings algorithm satisfies the detailed balance w.r.p. P.

  Thus if the Markov chain defined by T is regular, P is the unique stationary distribution.

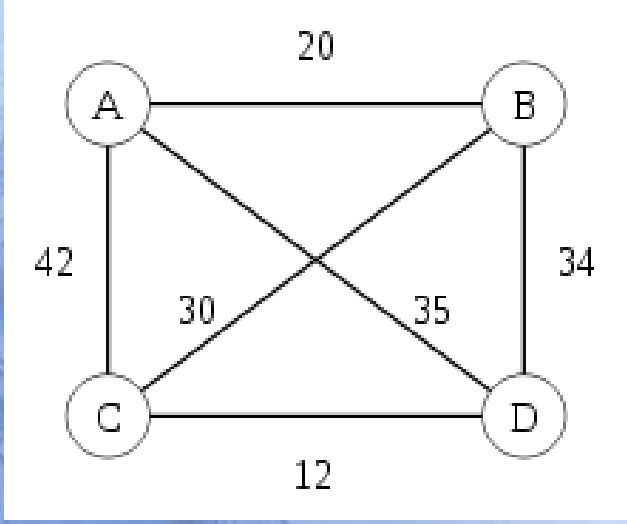
Convergence

- Mixing time for MCMC:
  - What we need is the stationary distribution of the Markov chain, but how long does it take to converge --- mixing time (burn in).
  - Gibbs sampling sometimes has very slow convergence.
  - Metropolis-Hastings's convergence depends on the proposal distribution.

- Theory for the convergence for MCMC:
  - For a Markov chain, the largest eigenvalue of the transition matrix T is 1; the **gap** between the largest and the second largest (in absolute value) eigenvalue determines the mixing time.
- A main challenge for PGM:
  - Design MCMC algorithms that: 1) efficiently implementable for PGMs; 2) mixing time is not too long.

Some Thoughts and Open Problems

- Inference is **NP**-hard, what shall we do?
  - Develop practical algorithms. **#P** complete is a worst case result.
  - To solve the inference problem, we are in a situation very similar to solving TSP (**NP**-hard):



TSP: Find the shortest path such that each vertex is visited exactly once.

- TSP (decision) is **NP**-complete.
- Euclidean TSP is NP-hard.
- Approximate TSP is NP-hard.

Arora proved (Gödel prize, 2010) :

Euclidean TSP + approximation

$\longrightarrow$ **polynomial** approximation scheme

- Can we find a reasonable class of graphical models such that (approximate) inference has **polynomial** time algorithm?

- 联系方式

  - Email: wanglw@cis.pku.edu.cn

  - 个人主页：http://www.cis.pku.edu.cn/faculty/vision/wangliwei/

Thanks!

http://www.cis.pku.edu.cn/faculty/vision/wangliwei/

wanglw@cis.pku.edu.cn