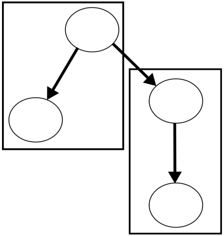
Relational Probabilistic Graphical Models

Probabilistic Graphical Models

L. Enrique Sucar, INAOE



Outline

Introduction

- The *standard* probabilistic graphical models have to represent explicitly each object in the domain, so they are equivalent in terms of their logical expressive power to propositional logic
- There are problems in which the number of objects (variables) could increase significantly, so a more expressive (compact) representation is desirable – i.e. model a student's knowledge of a certain topic (student modeling or, in general, user modeling), and that we want to include in the model all of the students in a college, where each student is enrolled in several topics
- It would be more efficient if in some way we could have a general model that represents the dependency relations for any student, S and any course, C, which could then be parameterized for particular cases

PRGMs

- Relational probabilistic graphical models (RPGMs) combine the expressive power of predicate logic with the uncertain reasoning capabilities of probabilistic graphical models
- Some of these models extend PGMs such as Bayesian networks or Markov networks by representing objects, their attributes, and their relations with other objects
- Other approaches extend the logic–based representations, in order to incorporate uncertainty, by describing a probability distribution over logic formulas

Types of PRGMs

- Extensions of logic models
  1. Undirected graphical models
     1. Markov Logic Networks
  2. Directed graphical models
     1. Bayesian Logic Programs
     2. Bayesian Logic Networks
- Extensions of probabilistic models
  1. Undirected graphical models
     1. Relational Markov Networks
     2. Relational Dependency Networks
     3. Conditional Random Fields
  2. Directed graphical models
     1. Relational Bayesian Networks
     2. Probabilistic Relational Models

Type of PRGMs

- Extensions of programming languages
  1. Stochastic Logic Programs
  2. Probabilistic Inductive Logic programming
  3. Bayesian Logic (BLOG)
  4. Probabilistic Modeling Language (IBAL)
- In this chapter we will review two of them:
  - *Probabilistic relational models* extend Bayesian networks to incorporate objects and relations, as in a relational data base
  - *Markov logic networks*, which add weights to to logical formulas, and can be considered as an extension of Markov networks

Probabilistic Relational Models

- Probabilistic relational models (PRMs) are an extension of Bayesian networks that provide a more expressive, object-oriented representation
- For the case of a very large model, only part of it is considered at any time, so the inference complexity is reduced

Representation

- The basic entities in a PRM are objects or domain entities, whcih are partitioned into a set of disjoint classes $X_1, ..., X_n$
- Each class is associated with a set of attributes $A(X_i)$
- Each attribute $A_{ij} \in A(X_i)$ (that is, attribute j of class i) takes on values in some fixed domain of values $V(A_{ij})$
- A set of relations, R_j, are defined between the classes
- The classes and relations define the *schema* of the model
- The dependency model is defined at the class level, allowing it to be used for any object in the class

Dependency model

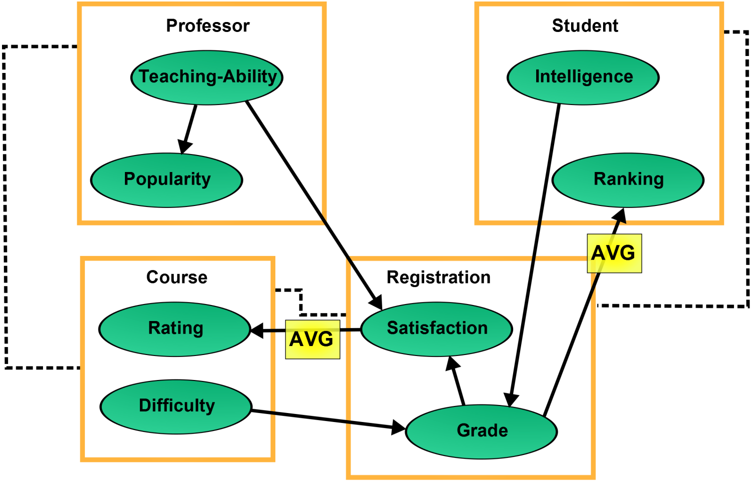
- A PRM specifies the probability distribution using the same underlying principles used in Bayesian networks
- Each of the random variables in a PRM, the attributes $x.a$ of the individual objects x, is directly influenced by other attributes, which are its parents
- A PRM, therefore, defines for each attribute, a set of parents, which are the directed influences on it, and a local probabilistic model that specifies probabilistic parameters
- There are two differences between PRMs and BNs: (i) In a PRM the dependency model is specified at the class level, allowing it to be used for any object in the class. (ii) A PRM explicitly uses the relational structure of the model, allowing an attribute of an object to depend on attributes of related objects

Example - school domain

- There are 4 classes, with 2 attributes each in this example:

Professor:	teaching-ability, popularity
Student:	intelligence, ranking
Course:	rating, difficulty
Registration:	satisfaction, grade

Example


Professor
Teaching-Ability
Popularity
Student
Intelligence
Ranking
AVG
Course
Rating
Difficulty
AVG
Registration
Satisfaction
Grade

Types of attributes

- This representation allows for two types of attributes in each class: (i) information variables, (ii) random variables.
- The random variables are the ones that are linked in a kind of Bayesian network that is called a *skeleton*
- From this skeleton, different Bayesian networks can be generated, according to other variables in the model
- This gives the model a greater flexibility and generality, facilitating knowledge acquisition
- It also makes inference more efficient, because only part of the model is used in each particular case

Parameters

- The probability distribution for the skeletons are specified as in Bayesian networks
- A PRM defines for each attribute $x.a$, a set of parents, which are the directed influences on it, and a local probabilistic model that specifies the conditional probability of the attribute given its parents
- To guarantee that the local models define a coherent global probability distribution, the underlying dependency structure should be acyclic, as in a BN

Inference

- The inference mechanism for PRMs is the same as for Bayesian networks, once the model is instantiated to particular objects in the domain
- PRMs can take advantage of two properties to make inference more efficient
- One property is the locality of influence, most attributes will tend to depend mostly on attributes of the same class, and there are few interclass dependencies – can take advantage of this locality property by using a divide and conquer approach
- The other aspect is reuse. In a PRM there are usually several objects of the same class, with similar structure and parameters. Once inference is performed for one object, this can be reused for the other similar objects

Learning

- Given that PRMs share the same underlying principles of BNs, the learning techniques developed for BNs can be extended for PRMs
- The expectation maximization algorithm has been extended to learn the parameters of a PRM, and structure learning techniques have been developed to learn the dependency structure from a relational database

Markov Logic Networks

- In contrast to PRMs, Markov logic networks (MLN) start from a logic representation, adding *weights* to formulas to incorporate uncertainty
- In logic, a $\mathcal{L}$-interpretation which violates a formula given in a knowledge base (KB) has zero probability
- In Markov Logic Networks, this assumption is relaxed. If the interpretation violates the KB, it has *less* probability than others with no violations
- In a MLN, a weight to each formula is added in order to reflect how strong the constraint is

Markov Networks - brief review

- A Markov Network is a model for the joint distribution of a set of variables $X = (X_1, X_2, ..., X_n) \in \mathcal{X}$
- It is composed of an undirected graph G with a node per variable, and a set of potential functions ϕ_k, one for each clique
- The joint distribution represented by a Markov network is given by

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{1}$$

Markov networks

- Markov Networks can also be represented using *log-linear* models, where each clique potential function is replaced by an exponential weighted sum:

$$P(X = x) = \frac{1}{z} \exp \sum_{j} w_j f_j(x) \tag{2}$$

- Where w_j is a weight (real value) and f_j is, for our purposes, a binary formula $f_j(x) \in \{0, 1\}$

MLN

- An MLN L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number
- Together with a finite set of constants $C = \{c_1, c_2, ..., c_{|C|}\}$, it defines a Markov network $M_{L,C}$:
  1. $M_{L,C}$ contains one binary node for each possible grounding of each formula appearing in the MLN L. The value of the node is 1 if the ground atom is true, and 0 otherwise.
  2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.

Ground MLN

- MLNs are a generalization of Markov networks, so they can be seen as templates for constructing Markov networks
- Given a MLN and a set of different constants, different Markov networks can be produced; these are known as ground Markov networks
- The joint probability distribution of a ground Markov network is defined in a similar way as a Markov network
- The graphical structure of a MLN is based on its definition; there is an edge between two nodes of the MLN if the corresponding ground atoms appear together in at least one grounding of one formula in the knowledge base

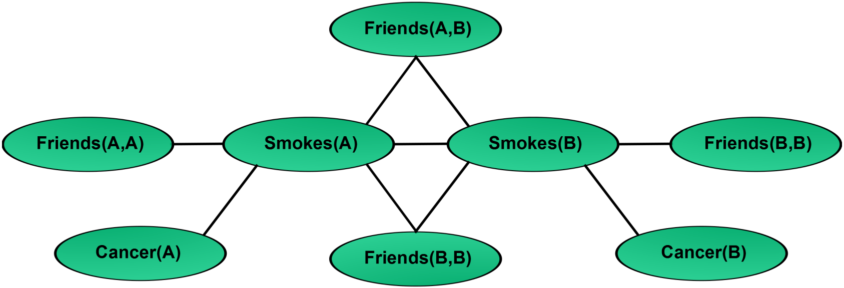
MLN - example

- MLN consisting of two logical formulas:

$$\forall x Smoking(x) \rightarrow Cancer(x)$$

$$\forall x \forall y Friends(x) \rightarrow (Smoking(x) \leftrightarrow Smoking(y))$$

- x and y are instantiated to the constants A and B



Inference

- Inference in MLN consists in estimating the probability of a logical formula, F_1, given that another formula (or formulas), F_2, are true
- That is, calculating $P(F_1 \mid F_2, L, C)$, where L is a MLN consisting of a set of weighted logical formulas, and C is a set of constants
- To compute this probability, we can estimate the proportion of possible worlds in which F_1 and F_2 are true, over the possible worlds in which F_2 holds – the probability of each possible world is considered according to the weights of the formulas and the structure of the grounded Markov network

Inference

- Performing the previous calculations directly is, computationally, very costly
- One alternative is using stochastic simulation; by sampling the possible worlds we can obtain an estimate of the desired probability; for instance, using the Markov chain Montecarlo techniques
- Another alternative is to make certain reasonable assumptions about the structure of the logical formulas that simplify the inference process – for instance that F_1 and F_2 are conjunctions of ground literals

Learning

- Learning a MLN involves two aspects. One aspect is learning the logical formulas (structure), and the other is learning the weights for each formula (parameters)
- For learning the logical formulas, we can apply techniques from the area of inductive logic programming (ILP). There are different approaches that can induce logical relations from data, considering some background knowledge
- The weights of the logical formulas can be learned from a relational database, the weight of a formula is proportional to its number of true groundings in the data with respect to its expectation according to the model (can also use sampling)

Applications

- A *general* student model for virtual laboratories based on PRMs
- MLNs for representing visual grammars for object recognition

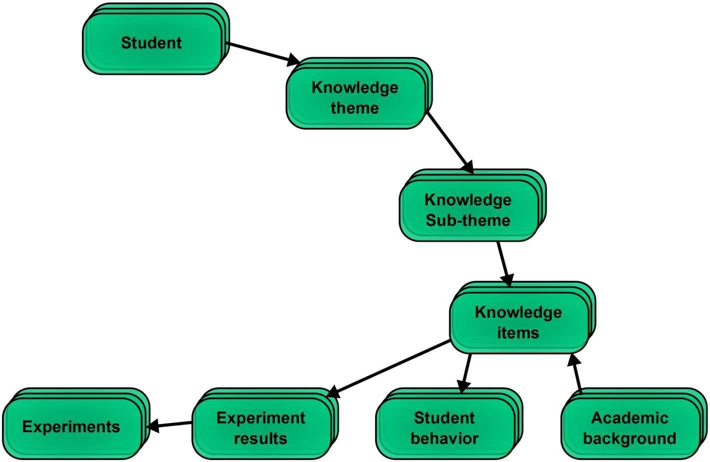
Student Modeling with PRMs

- A virtual lab provides a simulated model of some equipment, so that students can interact with it and learn by doing
- A tutor serves as a virtual assistant in this lab, providing help and advice to the user, and setting the difficulty of the experiments, according to the student's level
- The cognitive state should be obtained based solely on the student's interactions with the virtual lab and the results of the experiments – a student model
- The model infers, from the student's interactions with the laboratory, the cognitive state; and based on this, an intelligent tutor can give personalized advice to the student

Probabilistic relational student model

- PRMs provide a compact and natural representation for student modeling
- Each class represents the set of parameters of several students, like in databases, but the model also includes the probabilistic dependencies between classes for each student
- In order to apply PRMs to student modeling we have to define the main objects involved in the domain
- The dependency model is defined at the class level, allowing it to be used for any object in the class

High level model



Classes

Student: student-id, student-name, major, quarter, category.

Knowledge Theme: student-id, knowledge-theme-id, knowledge-theme-known.

Knowledge Sub-theme: student-id, knowledge-sub-theme-id, knowledge-sub-theme-known.

Knowledge Items: student-id, knowledge-item-id, knowledge-item-known.

Academic background: previous-course, grade.

Student behavior: student-id, experiment-id, behavior-var1, ...

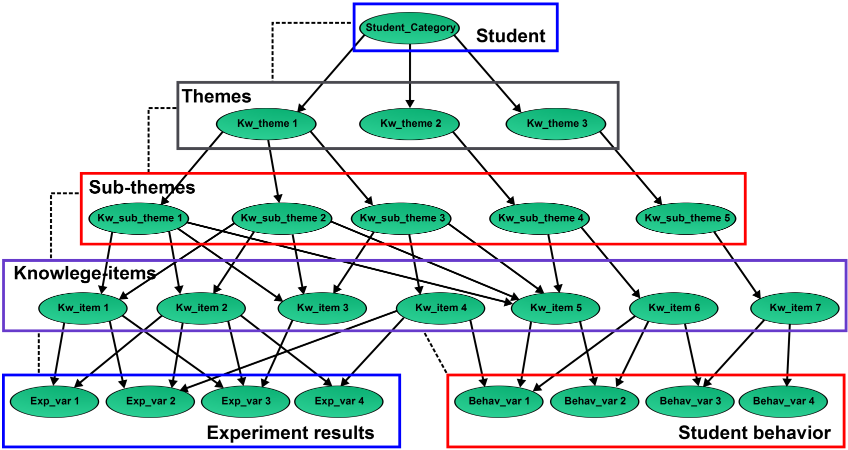
Experiment results: student-id, experiment-id, experiment-repetition, result-var1, ...

Experiments: experiment-id, experiment-description, exp-var1, exp-var2, ...

Skeleton

- From the PRM student model we can define a general Bayesian network, a skeleton, that can be instantiated for different scenarios, in this case experiments
- From the class model we obtain a hierarchical skeleton
- From the skeleton, it is possible to define different instances according to the values of specific variables in the model
- We can define particular instances for each experiment (for example, in the robotics domain, there could be experiments related to robot design, control, motion planning, etc.) and student level (novice, intermediate, advanced)

Skeleton


Student_Category
Student
Themes
Kw_theme 1
Kw_theme 2
Kw_theme 3
Sub-themes
Kw_sub_theme 1
Kw_sub_theme 2
Kw_sub_theme 3
Kw_sub_theme 4
Kw_sub_theme 5
Knowlege-items
Kw_item 1
Kw_item 2
Kw_item 3
Kw_item 4
Kw_item 5
Kw_item 6
Kw_item 7
Exp_var 1
Exp_var 2
Exp_var 3
Exp_var 4
Experiment results
Behav_var 1
Behav_var 2
Behav_var 3
Behav_var 4
Student behavior

Inference

- Once a specific Bayesian network is generated, it can be used to update the student model via standard probability propagation techniques
- In this case, it is used to propagate evidence from the experiment evaluation to the knowledge items, and to the knowledge sub-themes and to the knowledge themes
- This is used by the tutor to decide if it should provide help to the student, and at what level of detail
- For example, if in general the experiment was successful, but some aspect was not very good, a lesson on a specific concept (item) is given to the student

Visual Grammars

- A visual grammar describes objects hierarchically
- For visual object recognition, we need a grammar that allows us to model the decomposition of a visual object into its parts and how they relate with each other
- One interesting kind of relational grammar are Symbol-Relation Grammars

Object representation

- Classes of objects are represented based on symbol-relational grammars
- This includes three basic parts: (i) the basic elements of the grammar or lexicon, (ii) the spatial relations, (iii) the transformation rules
- The visual features considered are: *uniform color regions* (color is quantized in 32 levels) and edges at different orientations (obtained with Gabbor filters) – these are clustered and the centroids of these clusters constitute the *Visual Lexicon*

Relations

- The spatial relations include topological and order relationships
- The relationships used are: $Inside\_of(A, B)$ (A region is *within* B region), $Contains(A, B)$ (A region *covers* completely B region), $Left(A, B)$ (A is *touched* by B and A is located *left* from B), $Above(A, B)$ (A is *touched* by B and A is located *above* from B), $Invading(A, B)$ (A is *partially covering* B more than *Above* and *Left* but less than *Contains*)

Rules

- The next step is to generate the rules that make up the grammar
- Using training images for the class of the object of interest, the most common relationships between clusters are obtained – become candidate rules to build the grammar
- This is an iterative process where the rules are subsumed and converted to new non-terminal elements of the grammar
- The starting symbol of the grammar represents the class of objects to be recognized

Transforming a SR Grammar into a Markov Logic Network

- The SR grammar for a class of objects is transformed *directly* to formulas in the MLN language – structure of the model
- The parameters –weights associated to each formula–, are obtained from the training image set.

Example - MLN to recognize faces

- SR grammar to recognize faces based on high-level features: eyes, mouth, nose, head
- The productions of this simple SR-grammar for faces are:

$$
\begin{aligned}
1 : FACE^0 &\rightarrow < \{eyes^2, mouth^2\}, \{above(eyes^2, mouth^2)\} > \\
2 : FACE^0 &\rightarrow < \{nose^2, mouth^2\}, \{above(nose^2, mouth^2)\} > \\
3 : FACE^0 &\rightarrow < \{eyes^2, head^2\}, \{inside\_of(eyes^2, head^2)\} > \\
4 : FACE^0 &\rightarrow < \{nose^2, head^2\}, \{inside\_of(nose^2, head^2)\} > \\
5 : FACE^0 &\rightarrow < \{mouth^2, head^2\}, \{inside\_of(mouth^2, head^2)\}
\end{aligned}
$$

Transformation to MLN

First, we need to declare the formulas:

```
aboveEM(eyes,mouth)
aboveNM(nose,mouth)
insideOfEH(eyes,head)
insideOfNH(nose,head)
insideOfMH(mouth,head)
isFaceENMH(eyes,nose,mouth,head)
```

Subsequently, we need to declare the domain:

```
eyes={E1,E2,E3,E4}
nose={N1,N2,N3,N4}
mouth={M1,M2,M3,M4}
head={H1,H2,H3,H4}
```

Transformation to MLN

Finally we need to write the weighted first-order formulas. We used a validation image dataset and translated the probabilities into weights:

```
1.58 isFaceENMH(e,n,m,h) => aboveEM(e,m)
1.67 isFaceENMH(e,n,m,h) => aboveNM(n,m)
1.16 isFaceENMH(e,n,m,h) => insideOfEH(e,h)
1.25 isFaceENMH(e,n,m,h) => insideOfNH(n,h)
1.34 isFaceENMH(e,n,m,h) => insideOfMH(m,h)
```

Recognition

- To recognize a *face* in an image, the relevant aspects of the image are transformed into a first-order *KB*
- For this, the terminal elements are detected (in the example the eyes, mouth, nose and head) in the image, as well as the spatial relations between these elements
- Then, the particular image *KB* is combined with the *general* model represented as a MLN; and from this combination a *grounded Markov network* is generated
- Object (face) recognition is performed using standard probabilistic inference over the Markov network

Book

Sucar, L. E, *Probabilistic Graphical Models*, Springer 2015 – Chapter 12

Additional Reading (1)

- Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: L. Getoor, B. Taskar (eds.) Introduction to Statistical Relational Learning, pp. 339–371. MIT Press, Cambridge (2007)
- Ferrucci, F., Pacini, G., Satta, G., Sessa, M. I., Tortora, G., Tucci, M., Vitiello, G.: Symbol-Relation Grammars: A Formalism for Graphical Languages. Information and Computation, 131(1), 1-46 (1996)
- Friedman, N., Getoor, L., Koller, D., Pfeffe, A.: Learning Probabilistic Relational Models. In: Proceeding of the International Joint Conference on Artificial Intelligence (IJCAI), pp.1300–1309 (1999)
- Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann (1988)

Additional Reading (2)

- Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
- Koller D.: Probabilistic Relational Models. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, Lecture Notes in Artificial Intelligence 1634, Springer, 3–13 1999
- Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York (1994)
- Lemmon, E.J.: Beginning Logic. Hackett Publishing Company (1978)

Additional Reading (3)

- Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning. 62(1-2), 107–136 (2006)
- Ruiz, E., Sucar, L.E.: An Object Recognition Model Based on Visual Grammars and Bayesian Networks. In: Proceedings of the Pacific Rim Symposium on Image and Video Technology, LNCS 8333, pp. 349–359, Springer-Verlag (2014)
- Nienhuys-Cheng, S., de Wolf, R.: Foundations of Inductive Logic Programming. Springer-Verlag, Berlin (1991)
- Sucar, L.E., Noguez, J.: Student Modeling. In: O. Pourret, P. Naim, B. Marcot (eds.) Bayesian Belief Networks: A Practical Guide to Applications, pp. 173–186. Wiley and Sons (2008)