Relational Learning

CS 760: Machine Learning
Spring 2018
Mark Craven and David Page

www.biostat.wisc.edu/~craven/cs760

Goals for the lecture

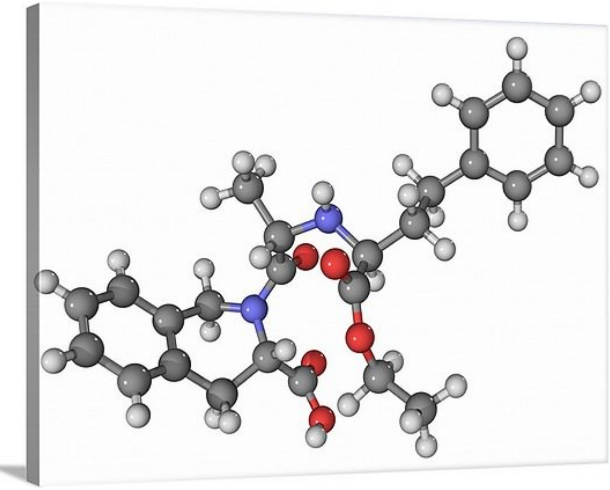
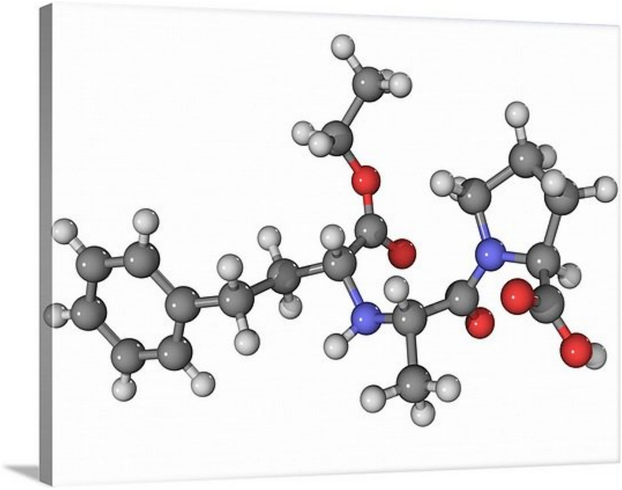
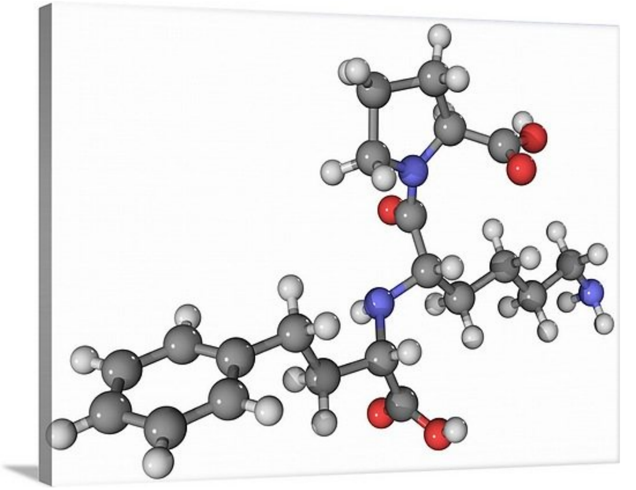
you should understand the following concepts

- relational learning
- the FOIL algorithm
- plate models

Relational learning example

consider the task of learning a *pharmacophore*: the substructure of a molecule that interacts with a target of interest

- instances for this task consist of interacting (+) and non-interacting molecules (-)

to represent each instance, we'd like to describe

- the (variable # of) atoms in the molecule
- the possible conformations of the molecule
- the bonds among atoms
- distances among atoms
- etc.

Relational learning example

[Finn et al., *Machine Learning* 1998]

a multi-relational representation for molecules

Molecule	Target_1	...	Target_n
mol1	inactive		i nactive
mol2	active		inactive
•			
•			
•			

Molecular Bioactivity

Molecule	Bond_ID	Atom_1_ID	Atom_2_ID	Bond_Type
mol1	bond1	a1	a2	aromatic
•				
•				
•				

Bonds

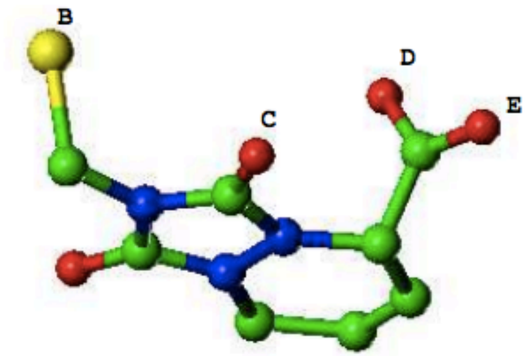
Molecule	Conformer	Atom_ID	Atom_Type	X_Coordinate	Y_Coordinate	Z_Coordinate
mol1	conf1	a1	carbon	2.58	−1.23	0.69
•						
•						
•						

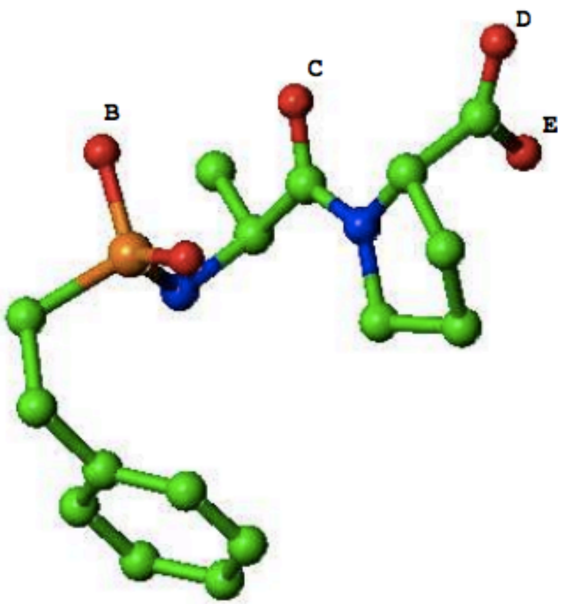
3D Atom Locations

Relational learning example

[Finn et al., *Machine Learning* 1998]

a learned relational rule characterizing ACE inhibitors






```
Molecule A is an ACE inhibitor if
for some conformer Conf of A:
   molecule A contains a zinc binding site B;
   molecule A contains a hydrogen acceptor C;
   the distance between B and C in Conf is 7.9 +/- .75;
   molecule A contains a hydrogen acceptor D;
   the distance between B and D in Conf is 8.5 +/- .75;
   the distance between C and D in Conf is 2.1 +/- .75;
   molecule A contains a hydrogen acceptor E;
   the distance between B and E in Conf is 4.9 +/- .75;
   the distance between C and E in Conf is 3.1 +/- .75;
   the distance between D and E in Conf is 3.8 +/- .75.
```

Relational representation

ACE_inhibitor(A) ← has_zinc_binding_site(A, B) ⋀
has_hydrogren_acceptor(A, C) ⋀
distance(B, C, 7.9, 0.75) ⋀
has_hydrogen_acceptor(A, D) ⋀
distance(B, D, 8.5, 0.75) ⋀
distance(C, D, 8.5, 0.75) ⋀
has_hydrogen_acceptor(A, E) ⋀
distance(B, E, 4.9, 0.75) ⋀
distance(C, E, 3.1, 0.75) ⋀
distance(D, E, 3.8, 0.75)

To learn an equivalent rule with a feature-vector learner, what features would we need to represent?

has_zinc_binding_site
has_hydrogen_acceptor
zinc_binding_site_and hydrogen_acceptor_distance
hydrogen_acceptor_hydrogen_acceptor_distance
…

can easily encode distance between a pair of atoms; but this pharmacophore has 4 important atoms with 6 relevant distances among them

Relational learning example

[Page et al., *AAAI* 2012]

- Data from electronic health records (EHRs) is being used to learn models for risk assessment, adverse event detection, etc.
- A patient's record is described by multiple tables in a relational DB

demographics

PatientID	Gender	Birthdate
P1	M	3/22/63

diagnoses

PatientID	Date	Physician	Symptoms	Diagnosis
P1	1/1/01	Smith	palpitations	hypoglycemic
P1	2/1/03	Jones	fever, aches	influenza

labs

PatientID	Date	Lab Test	Result
P1	1/1/01	blood glucose	42
P1	1/9/01	blood glucose	45

genetics

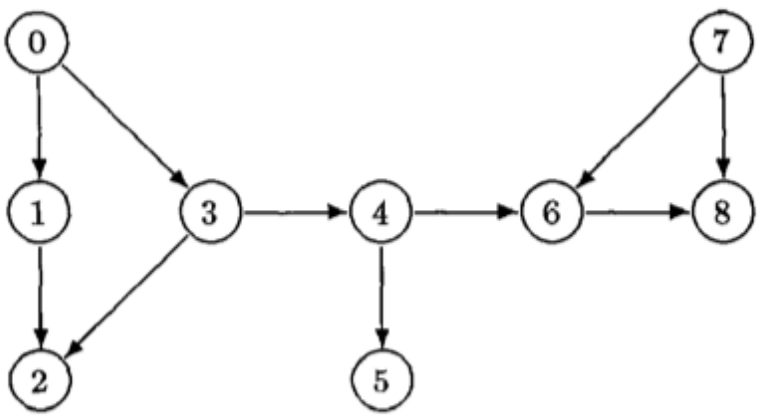
PatientID	SNP1	SNP2	…	SNP500K
P1	AA	AB		BB
P2	AB	BB		AA

drugs

PatientID	Date Prescribed	Date Filled	Physician	Medication	Dose	Duration
P1	5/17/98	5/18/98	Jones	prilosec	10mg	3 months

Relational learning example

- suppose we want to learn the general concept of can-reach in a graph, given a set of training instances describing a particular graph



⊕: ⟨0,1⟩ ⟨0,2⟩ ⟨0,3⟩ ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ ⟨1,2⟩ ⟨3,2⟩ ⟨3,4⟩ ⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ ⟨4,5⟩ ⟨4,6⟩ ⟨4,8⟩ ⟨6,8⟩ ⟨7,6⟩ ⟨7,8⟩

⊖: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩ ⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩ ⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩ ⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩ ⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩ ⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩ ⟨8,7⟩ ⟨8,8⟩

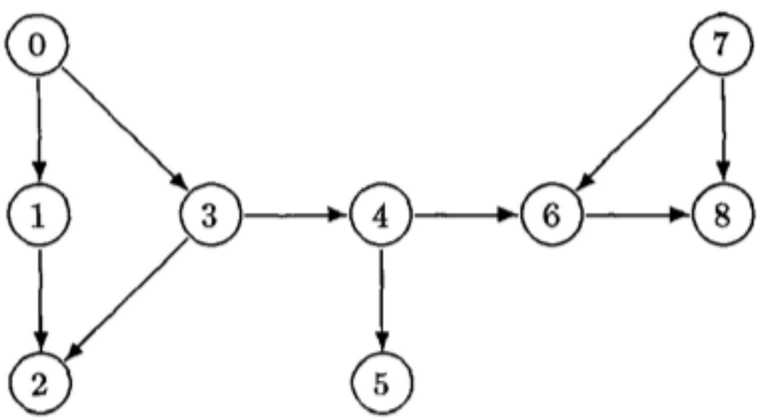
- how would you represent this task to a learner?

Relational learning example

- a relational representation, such as first-order logic, can capture this concept succinctly and in a general way

$$\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_2)$$

$$\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \wedge \text{can-reach}(X_3, X_2)$$



⊕: ⟨0,1⟩ ⟨0,2⟩ ⟨0,3⟩ ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ ⟨1,2⟩ ⟨3,2⟩ ⟨3,4⟩
⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ ⟨4,5⟩ ⟨4,6⟩ ⟨4,8⟩ ⟨6,8⟩ ⟨7,6⟩ ⟨7,8⟩

⊖: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩
⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩
⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩
⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩
⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩
⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩
⟨8,7⟩ ⟨8,8⟩

The FOIL algorithm for relational learning

[Quinlan, *Machine Learning* 1990]

given:

- tuples (instances) of a target relation
- *extensionally* represented background relations

do:

- learn a set of rules that (mostly) cover the positive tuples of the target relation, but not the negative tuples

Input to FOIL

- instances of target relation

$\oplus$: $\langle 0,1\rangle$ $\langle 0,2\rangle$ $\langle 0,3\rangle$ $\langle 0,4\rangle$ $\langle 0,5\rangle$ $\langle 0,6\rangle$ $\langle 0,8\rangle$ $\langle 1,2\rangle$ $\langle 3,2\rangle$ $\langle 3,4\rangle$
$\langle 3,5\rangle$ $\langle 3,6\rangle$ $\langle 3,8\rangle$ $\langle 4,5\rangle$ $\langle 4,6\rangle$ $\langle 4,8\rangle$ $\langle 6,8\rangle$ $\langle 7,6\rangle$ $\langle 7,8\rangle$

$\ominus$: $\langle 0,0\rangle$ $\langle 0,7\rangle$ $\langle 1,0\rangle$ $\langle 1,1\rangle$ $\langle 1,3\rangle$ $\langle 1,4\rangle$ $\langle 1,5\rangle$ $\langle 1,6\rangle$ $\langle 1,7\rangle$ $\langle 1,8\rangle$
$\langle 2,0\rangle$ $\langle 2,1\rangle$ $\langle 2,2\rangle$ $\langle 2,3\rangle$ $\langle 2,4\rangle$ $\langle 2,5\rangle$ $\langle 2,6\rangle$ $\langle 2,7\rangle$ $\langle 2,8\rangle$ $\langle 3,0\rangle$
$\langle 3,1\rangle$ $\langle 3,3\rangle$ $\langle 3,7\rangle$ $\langle 4,0\rangle$ $\langle 4,1\rangle$ $\langle 4,2\rangle$ $\langle 4,3\rangle$ $\langle 4,4\rangle$ $\langle 4,7\rangle$ $\langle 5,0\rangle$
$\langle 5,1\rangle$ $\langle 5,2\rangle$ $\langle 5,3\rangle$ $\langle 5,4\rangle$ $\langle 5,5\rangle$ $\langle 5,6\rangle$ $\langle 5,7\rangle$ $\langle 5,8\rangle$ $\langle 6,0\rangle$ $\langle 6,1\rangle$
$\langle 6,2\rangle$ $\langle 6,3\rangle$ $\langle 6,4\rangle$ $\langle 6,5\rangle$ $\langle 6,6\rangle$ $\langle 6,7\rangle$ $\langle 7,0\rangle$ $\langle 7,1\rangle$ $\langle 7,2\rangle$ $\langle 7,3\rangle$
$\langle 7,4\rangle$ $\langle 7,5\rangle$ $\langle 7,7\rangle$ $\langle 8,0\rangle$ $\langle 8,1\rangle$ $\langle 8,2\rangle$ $\langle 8,3\rangle$ $\langle 8,4\rangle$ $\langle 8,5\rangle$ $\langle 8,6\rangle$
$\langle 8,7\rangle$ $\langle 8,8\rangle$

- extensionally defined background relations

$$\textit{linked-to} = \{\langle 0,1\rangle, \langle 0,3\rangle, \langle 1,2\rangle, \langle 3,2\rangle, \langle 3,4\rangle, \langle 4,5\rangle, \langle 4,6\rangle, \langle 6,8\rangle, \langle 7,6\rangle, \langle 7,8\rangle\}$$

The FOIL algorithm for relational learning

FOIL uses a covering approach to learn a set of rules

LEARNRULESET(set of tuples T of target relation, background relations B)
{
$S = \{ \}$
repeat
$R \leftarrow$ LEARNRULE(T, B)
$S \leftarrow S \cup R$
$T \leftarrow T$ – positive tuples covered by R
until there are no (few) positive tuples left in T
return S
}

The FOIL algorithm for relational learning

LEARNRULE(set of tuples T of target relation, background relations B)
{
    $R = \{ \}$
    repeat
        $L \leftarrow$ best literal, based on T and B, to add to right-hand side of R
        $R \leftarrow R \cup L$
        $T \leftarrow$ new set of tuples that satisfy L
    until there are no (few) negative tuples left in T
    return R
}

Literals in FOIL

- Given the current rule $R(X_1, X_2, \ldots X_k) \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$
  FOIL considers adding several types of literals

$X_j = X_k$

$X_j \neq X_k$

both X_j and X_k either appear in the LHS of the rule, or were introduced by a previous literal

$Q(V_1, V_2, \ldots V_a)$

$\neg Q(V_1, V_2, \ldots V_a)$

at least one of the V_i's has to be in the LHS of the rule, or was introduced by a previous literal

where Q is a background relation

Literals in FOIL (continued)

$X_j = c$

where c is a constant

$X_j \neq c$

$X_j > a$

$X_j \leq a$

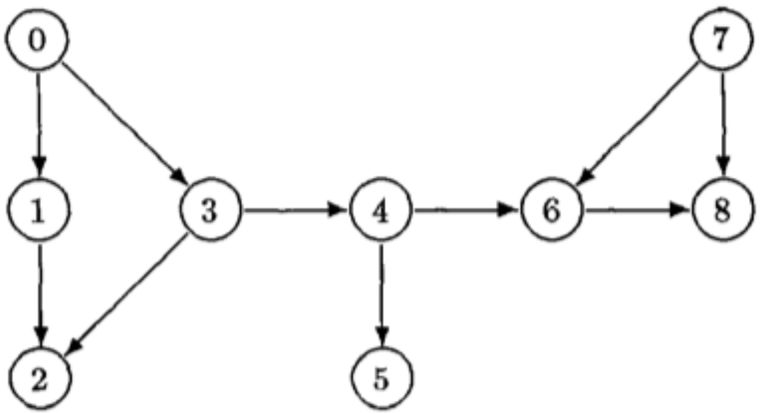
$X_j > X_k$

$X_j \leq X_k$

where X_j and X_k are numeric variables and a is a numeric constant

FOIL example

- suppose we want to learn rules for the target relation can-reach(X_1, X_2)
- we're given instances of the target relation from the following graph



$\oplus$: ⟨0,1⟩ ⟨0,2⟩ ⟨0,3⟩ ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ ⟨1,2⟩ ⟨3,2⟩ ⟨3,4⟩ ⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ ⟨4,5⟩ ⟨4,6⟩ ⟨4,8⟩ ⟨6,8⟩ ⟨7,6⟩ ⟨7,8⟩

$\ominus$: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩ ⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩ ⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩ ⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩ ⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩ ⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩ ⟨8,7⟩ ⟨8,8⟩

- and instances of the background relation linked-to

linked-to = {⟨0,1⟩, ⟨0,3⟩, ⟨1,2⟩, ⟨3,2⟩, ⟨3,4⟩, ⟨4,5⟩, ⟨4,6⟩, ⟨6,8⟩, ⟨7,6⟩, ⟨7,8⟩}

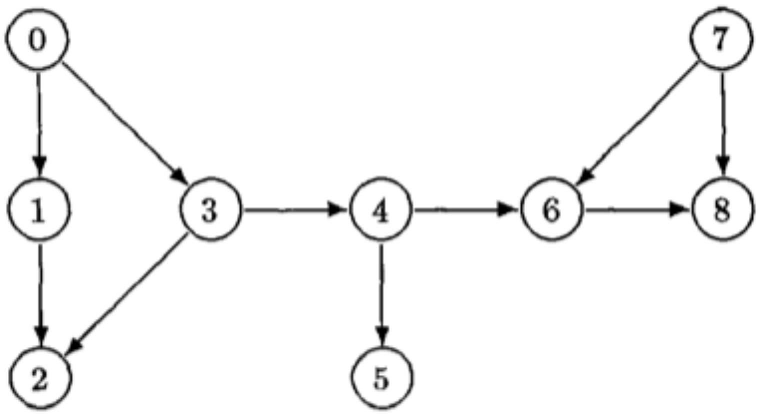
FOIL example

- the first rule learned covers 10 of the positive instances

  can-reach(X_1, X_2) ← linked-to(X_1, X_2)

- the second rule learned covers the other 9 positive instances

  can-reach(X_1, X_2) ← linked-to(X_1, X_3) $\wedge$ can-reach(X_3, X_2)



⊕: ⟨0,1⟩ ⟨0,2⟩ ⟨0,3⟩ ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ ⟨1,2⟩ ⟨3,2⟩ ⟨3,4⟩ ⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ ⟨4,5⟩ ⟨4,6⟩ ⟨4,8⟩ ⟨6,8⟩ ⟨7,6⟩ ⟨7,8⟩

⊖: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩ ⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩ ⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩ ⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩ ⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩ ⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩ ⟨8,7⟩ ⟨8,8⟩

- note that these rules generalize to other graphs

Evaluating literals in FOIL

- FOIL evaluates the addition of a literal L to a rule R by

$$FOIL\_Gain(L,R) = t\left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0}\right)$$

- where

  p_0 = # of positive tuples covered by R

  n_0 = # of negative tuples covered by R

  p_1 = # of positive tuples covered by $R \wedge L$

  n_1 = # of negative tuples covered by $R \wedge L$

  t = # of positive of tuples of R also covered by $R \wedge L$

- like information gain, but takes into account
  - we want to cover positives, not just get a more “pure” set of tuples
  - the size of the tuple set grows as we add new variables

Evaluating literals in FOIL

$$FOIL\_Gain(L,R) = t\left(Info\left(R_0\right) - Info\left(R_1\right)\right)$$

- where R_0 represents the rule without L and R_1 is the rule with L added
- $Info(R_i)$ is the number of bits required to encode a positive in the set of tuples covered by R_i

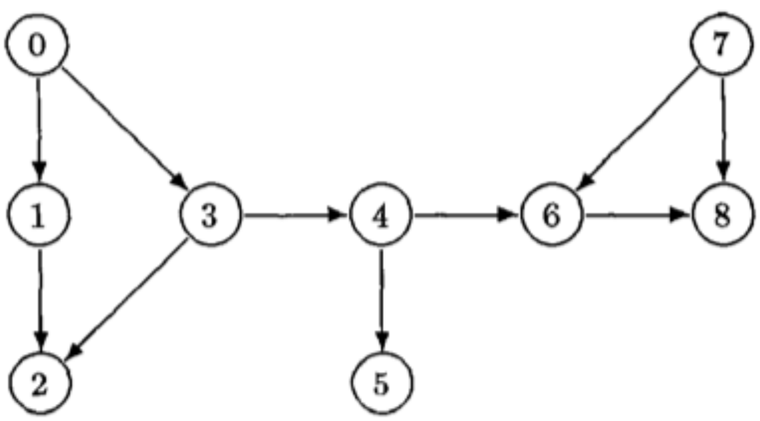
$$Info\left(R_i\right) = -\log_2\left(\frac{p_i}{p_i + n_i}\right)$$

Recall this example

- Definition of can-reach:

$$\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_2)$$

$$\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \wedge \text{can-reach}(X_3, X_2)$$



$\oplus$: ⟨0,1⟩ ⟨0,2⟩ ⟨0,3⟩ ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ ⟨1,2⟩ ⟨3,2⟩ ⟨3,4⟩ ⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ ⟨4,5⟩ ⟨4,6⟩ ⟨4,8⟩ ⟨6,8⟩ ⟨7,6⟩ ⟨7,8⟩

$\ominus$: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩ ⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩ ⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩ ⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩ ⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩ ⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩ ⟨8,7⟩ ⟨8,8⟩

FOIL example

- consider the first step in learning the second clause

can-reach(X_1, X_2) ←

⬇

can-reach(X_1, X_2) ←
linked-to(X_1, X_3)

$$FOIL\_Gain(L,R) = 9\left(\log_2 \frac{18}{18+54} - \log_2 \frac{9}{9+62}\right)$$
$$= 8.8$$

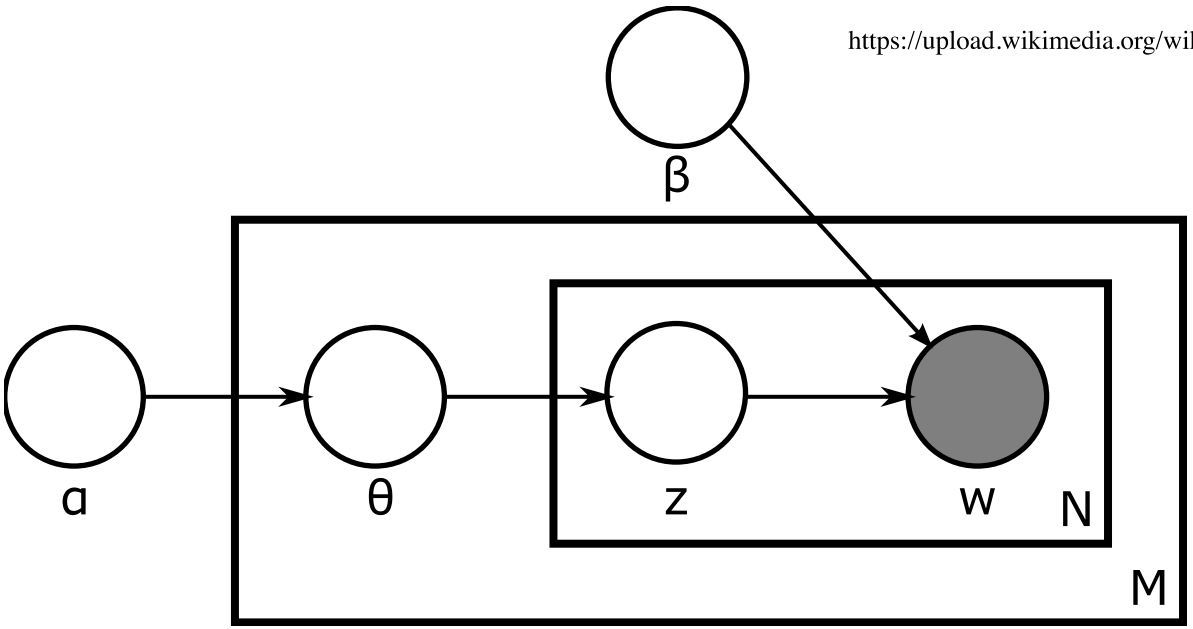
⊕: [redacted] ⟨0,2⟩ [redacted] ⟨0,4⟩ ⟨0,5⟩ ⟨0,6⟩ ⟨0,8⟩ [redacted] [redacted] [redacted]
⟨3,5⟩ ⟨3,6⟩ ⟨3,8⟩ [redacted] [redacted] ⟨4,8⟩ [redacted] [redacted] [redacted]

⊖: ⟨0,0⟩ ⟨0,7⟩ ⟨1,0⟩ ⟨1,1⟩ ⟨1,3⟩ ⟨1,4⟩ ⟨1,5⟩ ⟨1,6⟩ ⟨1,7⟩ ⟨1,8⟩
⟨2,0⟩ ⟨2,1⟩ ⟨2,2⟩ ⟨2,3⟩ ⟨2,4⟩ ⟨2,5⟩ ⟨2,6⟩ ⟨2,7⟩ ⟨2,8⟩ ⟨3,0⟩
⟨3,1⟩ ⟨3,3⟩ ⟨3,7⟩ ⟨4,0⟩ ⟨4,1⟩ ⟨4,2⟩ ⟨4,3⟩ ⟨4,4⟩ ⟨4,7⟩ ⟨5,0⟩
⟨5,1⟩ ⟨5,2⟩ ⟨5,3⟩ ⟨5,4⟩ ⟨5,5⟩ ⟨5,6⟩ ⟨5,7⟩ ⟨5,8⟩ ⟨6,0⟩ ⟨6,1⟩
⟨6,2⟩ ⟨6,3⟩ ⟨6,4⟩ ⟨6,5⟩ ⟨6,6⟩ ⟨6,7⟩ ⟨7,0⟩ ⟨7,1⟩ ⟨7,2⟩ ⟨7,3⟩
⟨7,4⟩ ⟨7,5⟩ ⟨7,7⟩ ⟨8,0⟩ ⟨8,1⟩ ⟨8,2⟩ ⟨8,3⟩ ⟨8,4⟩ ⟨8,5⟩ ⟨8,6⟩
⟨8,7⟩ ⟨8,8⟩

⊕: ⟨0,2,1⟩ ⟨0,2,3⟩ ⟨0,4,1⟩ ⟨0,4,3⟩ ⟨0,5,1⟩ ⟨0,5,3⟩ ⟨0,6,1⟩
⟨0,6,3⟩ ⟨0,8,1⟩ ⟨0,8,3⟩ ⟨3,5,2⟩ ⟨3,5,4⟩ ⟨3,6,2⟩ ⟨3,6,4⟩
⟨3,8,2⟩ ⟨3,8,4⟩ ⟨4,8,5⟩ ⟨4,8,6⟩

⊖: ⟨0,0,1⟩ ⟨0,0,3⟩ ⟨0,7,1⟩ ⟨0,7,3⟩ ⟨1,0,2⟩ ⟨1,1,2⟩ ⟨1,3,2⟩
⟨1,4,2⟩ ⟨1,5,2⟩ ⟨1,6,2⟩ ⟨1,7,2⟩ ⟨1,8,2⟩ ⟨3,0,2⟩ ⟨3,0,4⟩
⟨3,1,2⟩ ⟨3,1,4⟩ ⟨3,3,2⟩ ⟨3,3,4⟩ ⟨3,7,2⟩ ⟨3,7,4⟩ ⟨4,0,5⟩
⟨4,0,6⟩ ⟨4,1,5⟩ ⟨4,1,6⟩ ⟨4,2,5⟩ ⟨4,2,6⟩ ⟨4,3,5⟩ ⟨4,3,6⟩
⟨4,4,5⟩ ⟨4,4,6⟩ ⟨4,7,5⟩ ⟨4,7,6⟩ ⟨6,0,8⟩ ⟨6,1,8⟩ ⟨6,2,8⟩
⟨6,3,8⟩ ⟨6,4,8⟩ ⟨6,5,8⟩ ⟨6,6,8⟩ ⟨6,7,8⟩ ⟨7,0,6⟩ ⟨7,0,8⟩
⟨7,1,6⟩ ⟨7,1,8⟩ ⟨7,2,6⟩ ⟨7,2,8⟩ ⟨7,3,6⟩ ⟨7,3,8⟩ ⟨7,4,6⟩
⟨7,4,8⟩ ⟨7,5,6⟩ ⟨7,5,8⟩ ⟨7,7,6⟩ ⟨7,7,8⟩

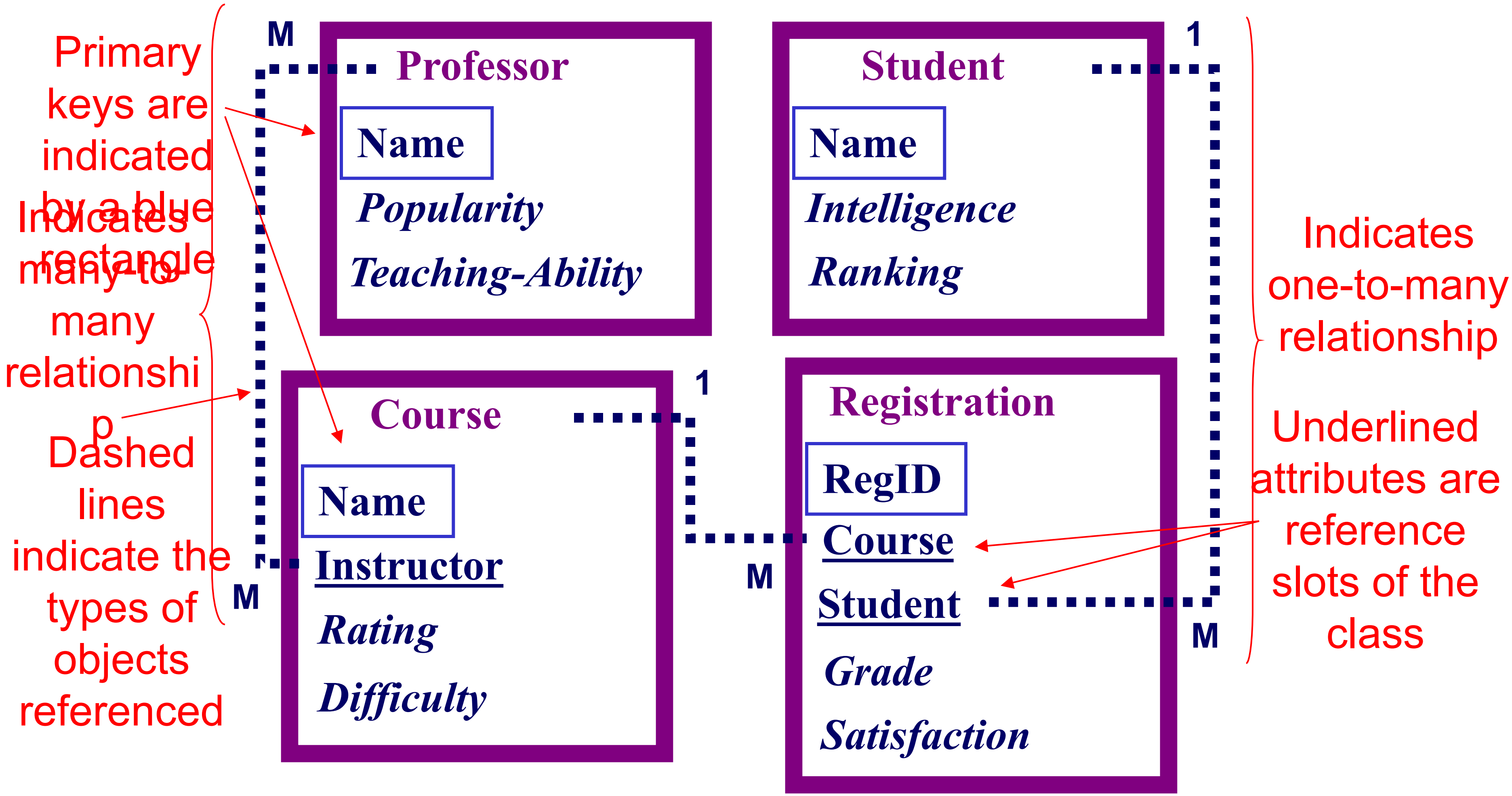
Alternative: Plates



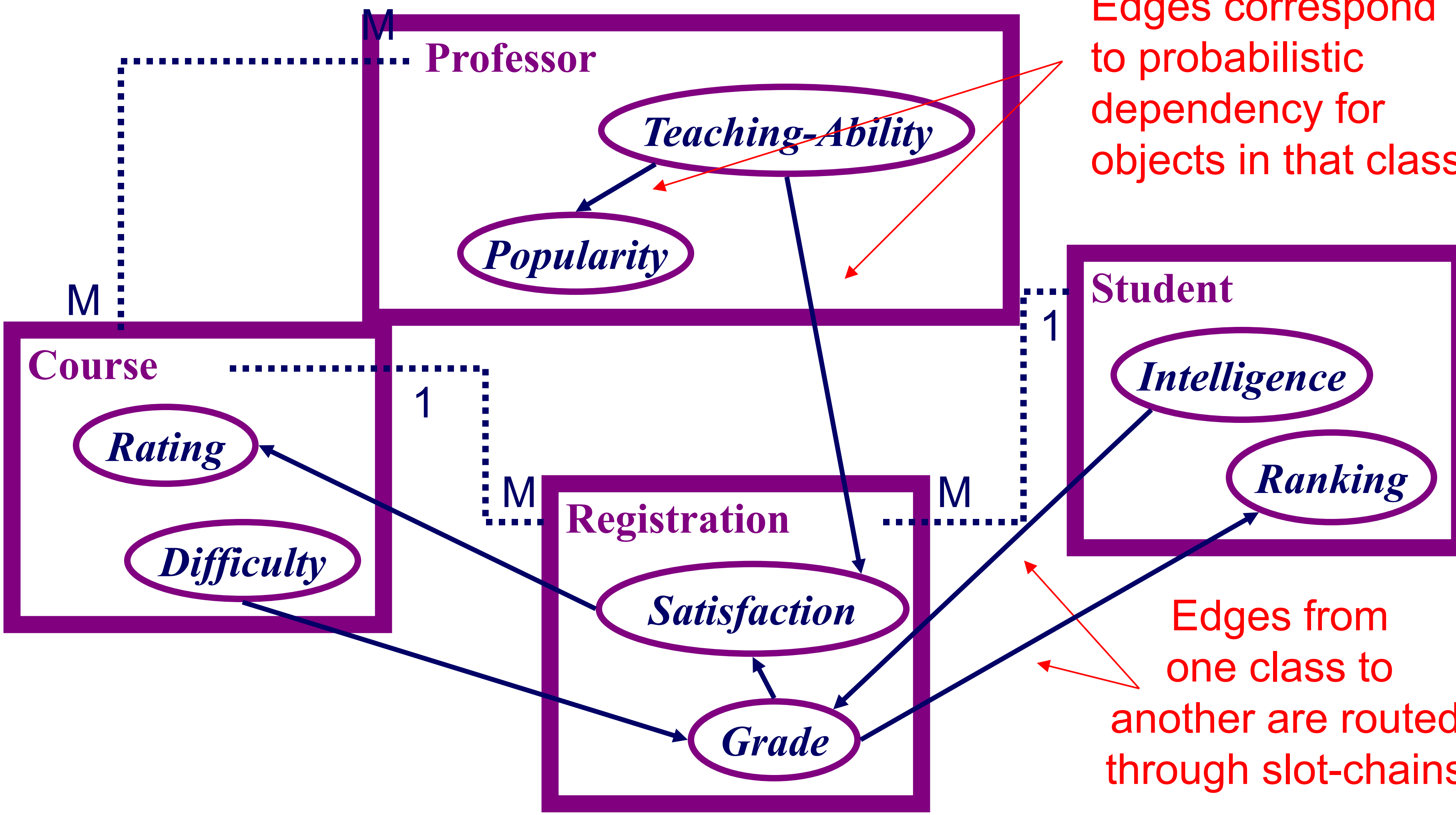
https://upload.wikimedia.org/wikipedia/commons/d/d3/Latent_Dirich...

A rectangle labeled "N" denotes N copies of the Bayes net inside.
Typically arcs go *into* rectangles, but we relax to allow outgoing next…

Alternative: Probabilistic Relational Models (PRMs)



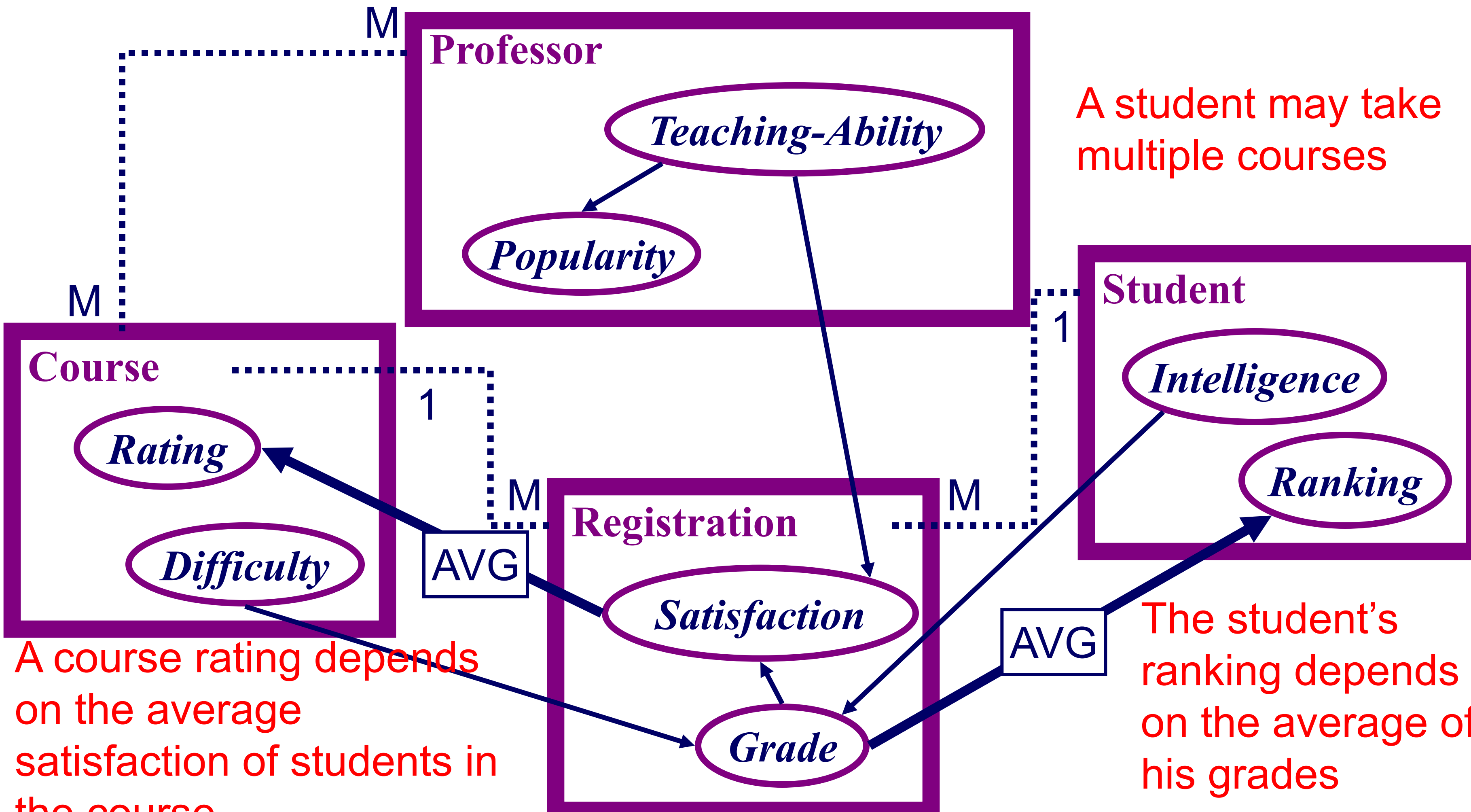
PRM Dependency Structure for the University Domain



Edges correspond to probabilistic dependency for objects in that class

Edges from one class to another are routed through slot-chains

PRM Dependency Structure

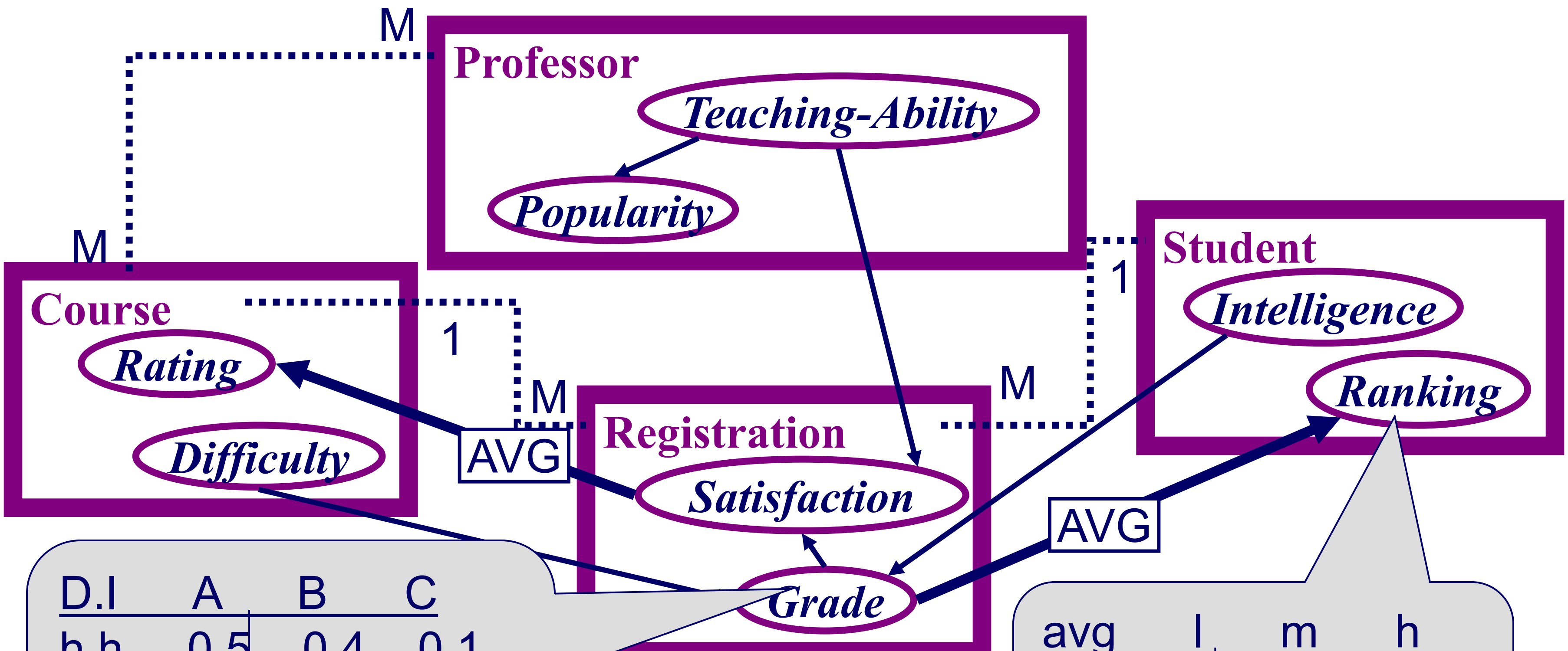


A student may take multiple courses

A course rating depends on the average satisfaction of students in the course

The student's ranking depends on the average of his grades

CPDs in PRMs



D.I	A	B	C
h,h	0.5	0.4	0.1
h,l	0.1	0.5	0.4
l,h	0.8	0.1	0.1
l,l	0.3	0.6	0.1

avg	l	m	h
A	0.1	0.2	0.7
B	0.2	0.4	0.4
C	0.6	0.3	0.1

Alternative: Markov logic

- a logical knowledge base is a set of **hard constraints** on the set of possible worlds

- let's make them **soft constraints**: when a world violates a formula, it becomes less probable, not impossible

- give each formula a weight (higher weight $\rightarrow$ stronger constraint)

$$\mathrm{P(world)} \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

MLN definition

- a *Markov Logic Network* (MLN) is a set of pairs (F, w) where
  F is a formula in first-order logic
  w is a real number

- together with a set of constants, it defines a Markov network with
  - one node for each grounding of each predicate in the MLN
  - one feature for each grounding of each formula F in the MLN, with the corresponding weight w

MLN example: friends & smokers

Smoking causes cancer.

Friends have similar smoking habits.

MLN example: friends & smokers

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$
$$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y) \right)$$

MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

MLN example: friends & smokers

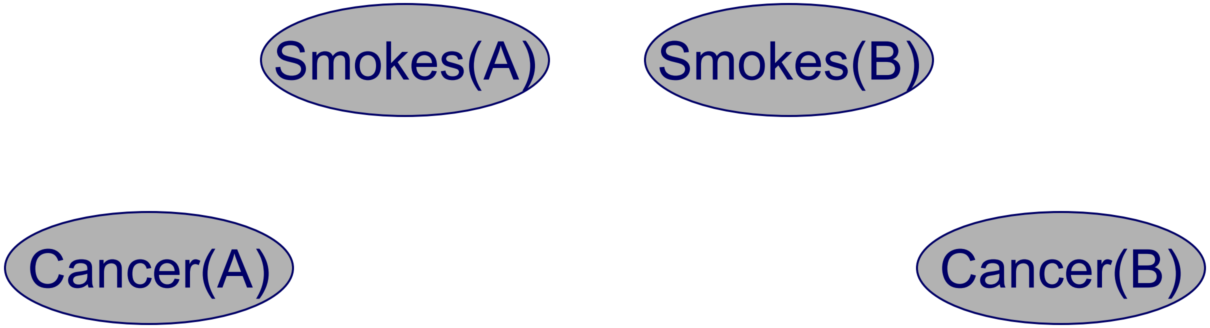
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)

MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

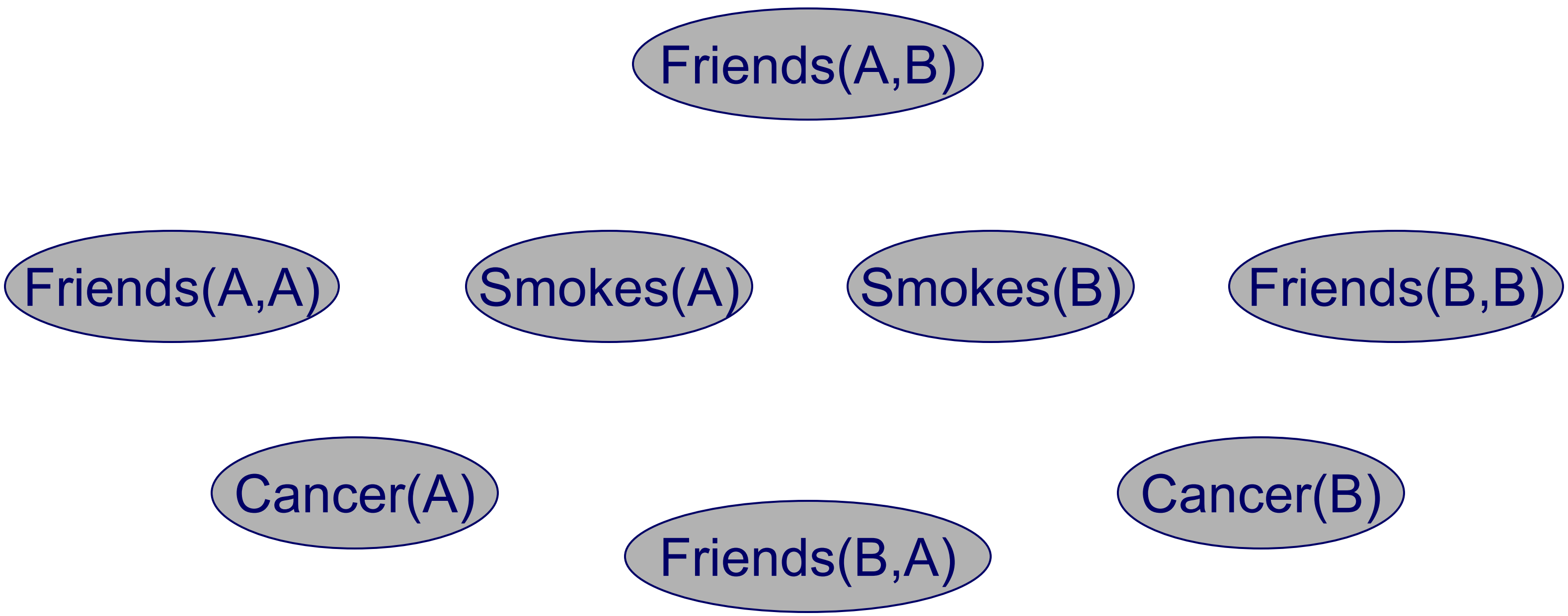
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

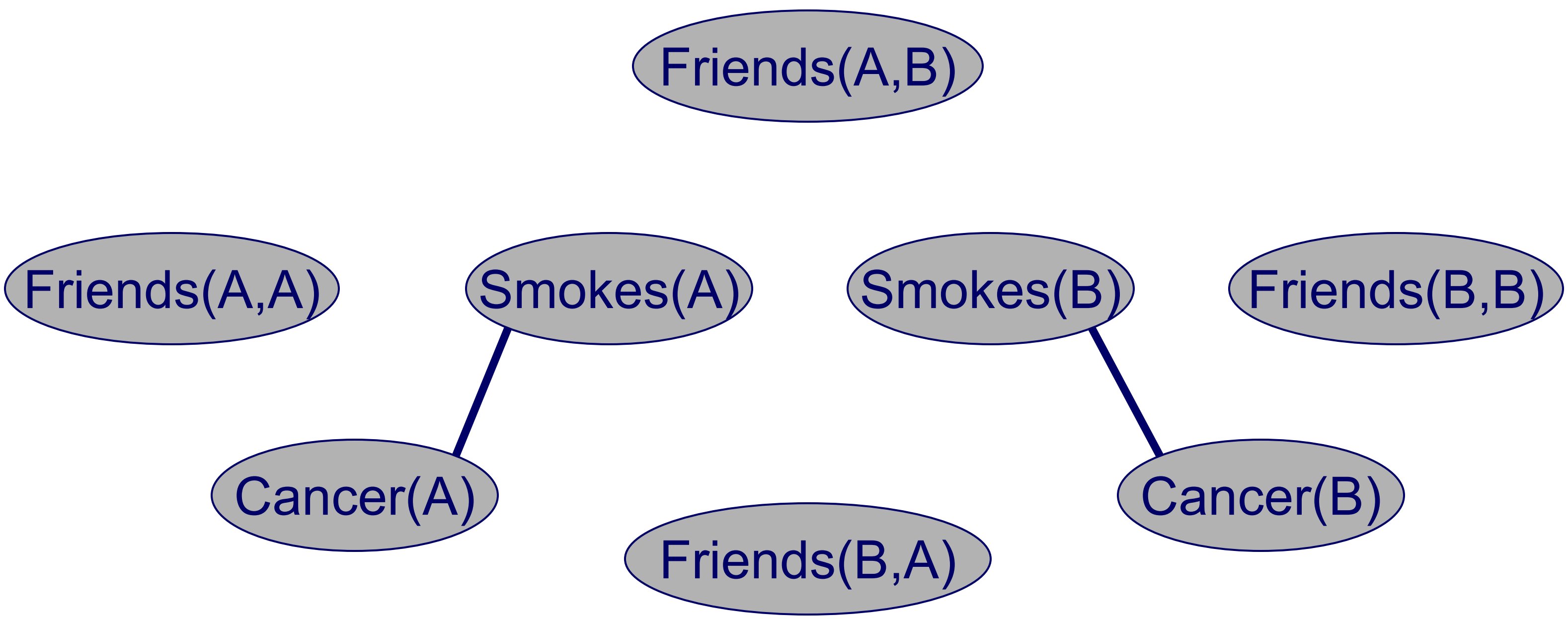
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

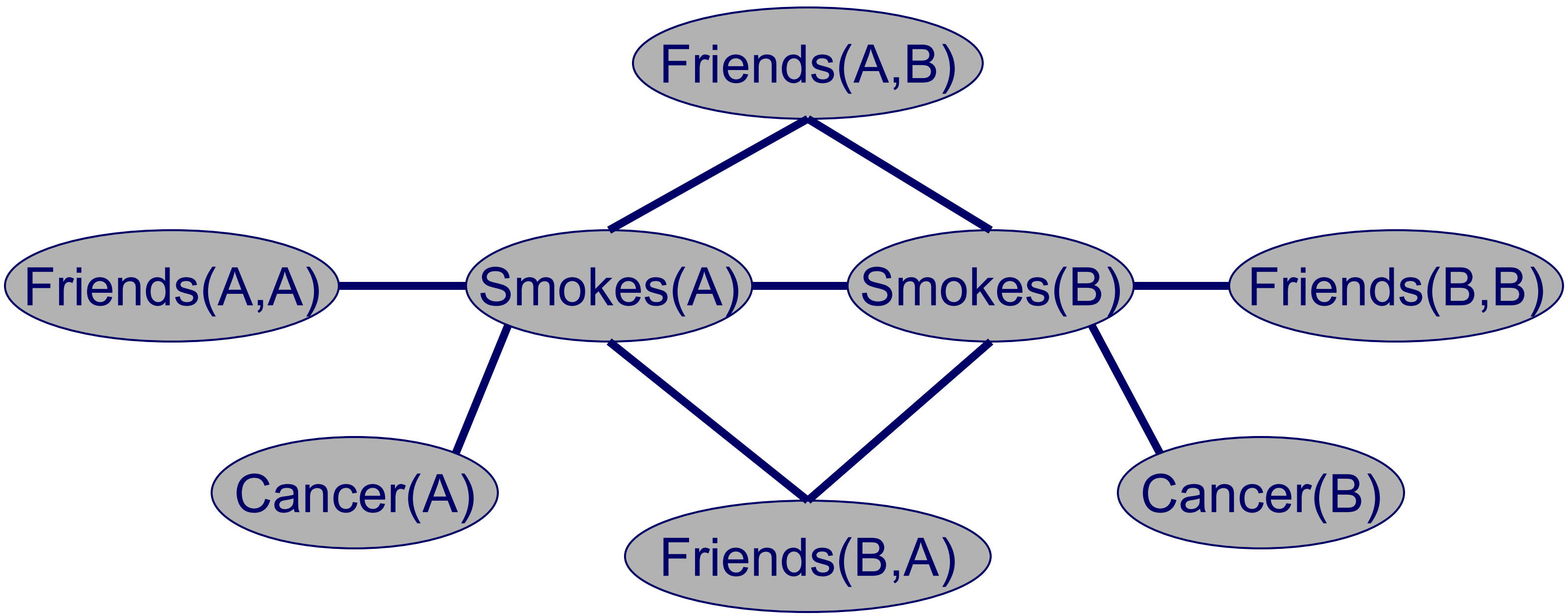
Two constants: **Anna** (A) and **Bob** (B)



MLN example: friends & smokers

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)



Markov logic networks

- a MLN is a **template** for ground Markov nets
  - the logic determines the form of the cliques
  - but if we had one more constant (say, **Larry**), we'd get a different Markov net

- we can determine the probability of a world $\boldsymbol{v}$ (assignment of truth values to ground predicates) by

$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(\boldsymbol{v})\right)$$

weight of formula i

# of true groundings of formula i in $\boldsymbol{v}$

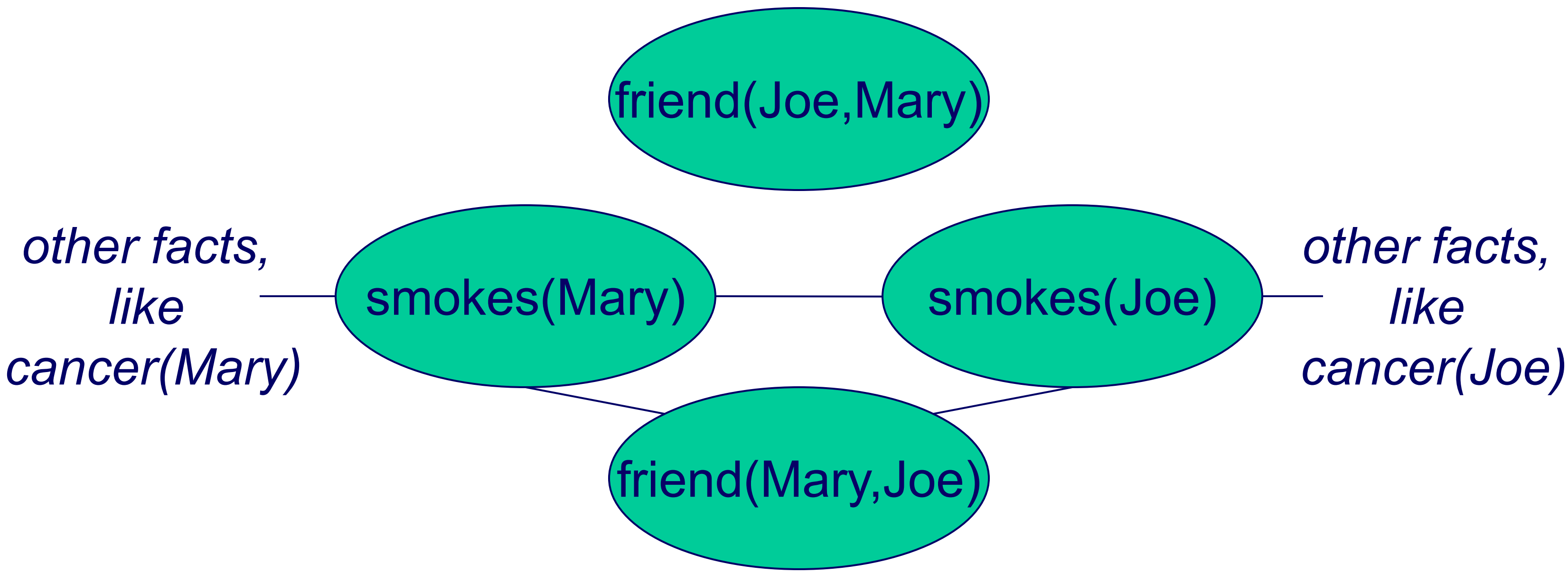
Example translated to Markov Network

- Facts:

smokes(Mary)
smokes(Joe)
friend(Joe, Mary)

- Rules:

Friend(x,y) ∧ Smokes(x) -> Smokes(y)



Computing weights

Consider the effect of rule R: Friends(x,y) ∧ Smokes(x) -> Smokes(y)

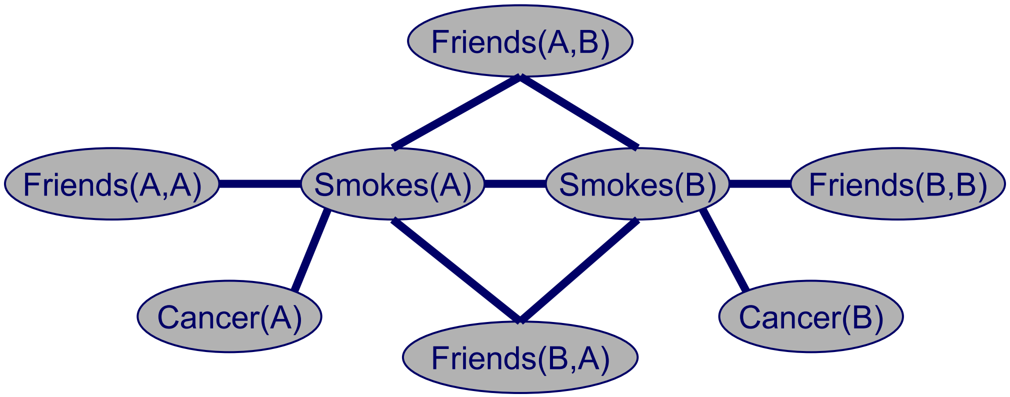
weight
1.1

	smokes(Mary)		¬smokes(Mary)	
	smokes(Joe)	¬smokes(Joe)	smokes(Joe)	¬smokes(Joe)
friends(Mary,Joe)	$e^{1.1}$	1	$e^{1.1}$	$e^{1.1}$
¬friends(Mary,Joe)	$e^{1.1}$	$e^{1.1}$	$e^{1.1}$	$e^{1.1}$

This is the only setting that does not satisfy R
Thus, it is given value 1, while the others are
Given value exp(weight(R))

Probability of a world in an MLN

$$\boldsymbol{v} = \begin{bmatrix} \text{Friends(A,A)} = \text{T} \\ \text{Friends(A,B)} = \text{T} \\ \text{Friends(B,A)} = \text{T} \\ \text{Friends(B,B)} = \text{T} \\ \text{Smokes(A)} = \text{F} \\ \text{Smokes(B)} = \text{T} \\ \text{Cancer(A)} = \text{F} \\ \text{Cancer(B)} = \text{F} \end{bmatrix}$$



$\forall x\ Smokes(x) \Rightarrow Cancer(x)$

$x = \text{A}$ T

$x = \text{B}$ F

$n_1(\boldsymbol{v}) = 1$

$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

$x = \text{A}, y = \text{A}$ T

$x = \text{A}, y = \text{B}$ F

$x = \text{B}, y = \text{A}$ F

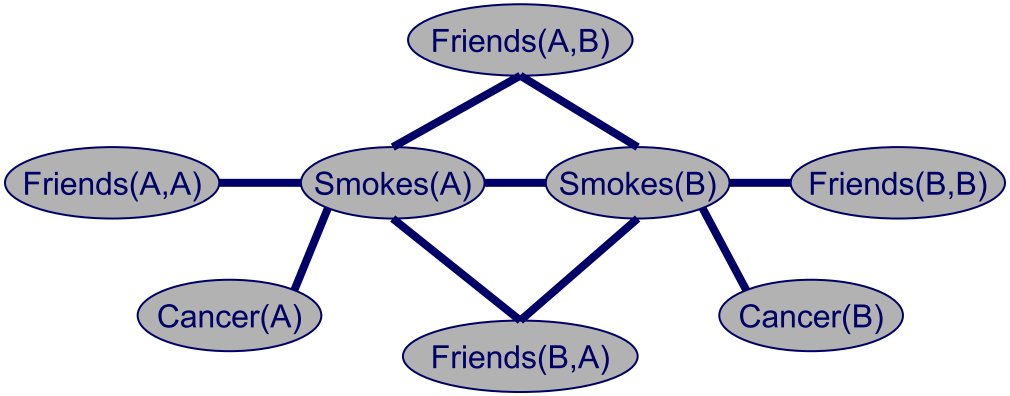
$x = \text{B}, y = \text{B}$ T

$n_2(\boldsymbol{v}) = 2$

# of true groundings of formula 1 in $\boldsymbol{v}$

Probability of a world in an MLN

$$\boldsymbol{v} = \begin{bmatrix} \text{Friends(A,A)} = \text{T} \\ \text{Friends(A,B)} = \text{T} \\ \text{Friends(B,A)} = \text{T} \\ \text{Friends(B,B)} = \text{T} \\ \text{Smokes(A)} = \text{F} \\ \text{Smokes(B)} = \text{T} \\ \text{Cancer(A)} = \text{F} \\ \text{Cancer(B)} = \text{F} \end{bmatrix}$$



$$P(\boldsymbol{v}) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(\boldsymbol{v})\right)$$

$$= \frac{1}{Z}\exp\left(1.5\left(1\right) + 1.1\left(2\right)\right)$$

Three MLN tasks

- inference: can use the toolbox of inference methods developed for ordinary Markov networks
  - Markov chain Monte Carlo methods, including persistent contrastive divergence (PCD) where we iterate MCMC and gradient ascent steps (don't wait for MCMC to converge)
  - belief propagation
  - variational methods

  in tandem with weighted SAT solver
  (e.g., MaxWalkSAT [Kautz et al., 1997])

- parameter learning

- structure learning: can use ordinary relational learning methods like FOIL or other ILP algorithms to learn new formula

MLN learning tasks

- the input to the learning process is a relational database of ground atoms

Friends(x, y)	Smokes(x)	Cancer(x)
Anna, Anna	Bob	Bob
Anna, Bob		
Bob, Anna		
Bob, Bob		

- the closed world assumption is used to infer the truth values of atoms not present in the DB

Parameter learning

- parameters (weights on formulas) can be learned using gradient ascent

$$\frac{\partial}{\partial w_i} \log P_w(\boldsymbol{v}) = n_i(\boldsymbol{v}) - E_w\left[n_i(\boldsymbol{v})\right]$$

# of times clause i is true in data

Expected # times clause i is true according to MLN

- approximation methods may be needed to estimate both terms

MLN experiment

- testbed: a DB describing Univ. of Washington CS department
  - 12 predicates

    Professor(*person*)

    Student(*person*)

    Area(*x*, *area*)

    AuthorOf(*publication*, *person*)

    AdvisedBy(*person*, *person*)

    etc.

  - 2707 constants

    publication (342)

    person (442)

    course (176)

    project (153)

    etc.

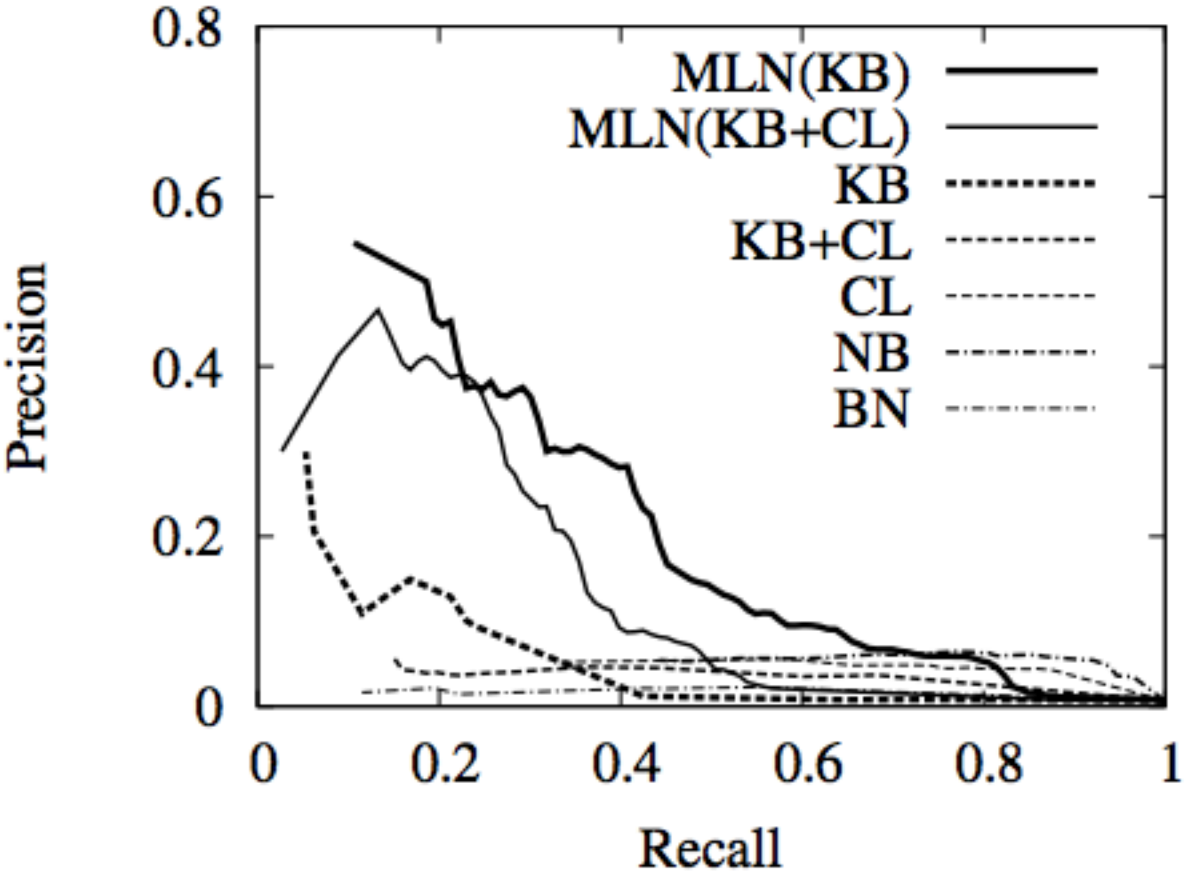
MLN experiment

- obtained knowledge base by having four subjects provide a set of formulas in first-order logic describing the domain

- the formulas in the KB represent statements such as
  - students are not professors
  - each student has at most one advisor
  - if a student is an author of a paper, so is her advisor
  - at most one author of a given publication is a professor
  - etc.

- note that the KB is not consistent

Learning to predict the AdvisedBy(x, y) relation



MLN w/ original KB
MLN w/ KB + ILP learned rule
KB alone
KB + ILP learned rules
ILP learned rules
naïve Bayes
Bayes net learner