Bayesian Learning

Ramón Béjar

Departament d'Informàtica
Universitat de Lleida

Learning Bayesian Models

How can we learn a Bayesian model from a database of past cases ?

We consider two basic problems:

1. Parameter learning: We have an structure for the Bayesian Network, but we miss the conditional probabilities associated with the factorization
2. Structure learning: We have neither the structure nor the conditional probabilities

Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE): Given a model BN and database of cases D, what are the most probable parameters of the corresponding probability tables for BN ?

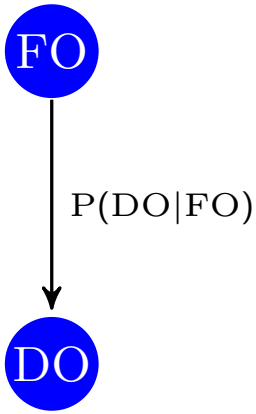
For a BN with n variables, we need to estimate n probability tables

The probability table for a variable x_i where the total number of cases to be considered with its set of parent variables is q_i has

$$(r_i - 1)q_i$$

independent probability parameters

Maximum Likelihood Estimation

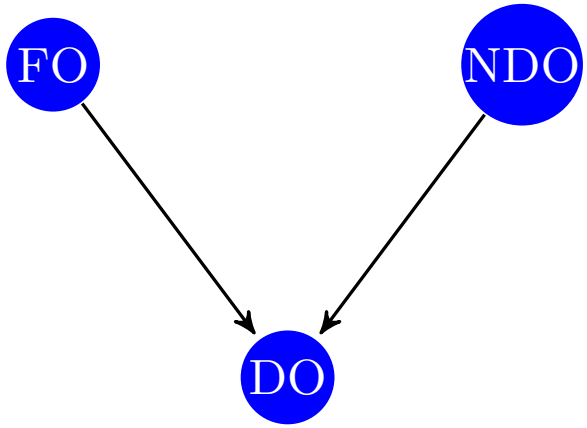


Factorization:

$$P(DO|FO)P(FO)$$

Independent parameters:

- For P(FO) : $\{P(FO=0)\}$
- For P(DO|FO) : $\{P(DO=0|FO=0), P(DO=0|FO=1)\}$



Factorization:

$$P(DO|FO,NDO)P(FO)P(NDO)$$

Independent parameters:

- For P(FO) : $\{P(FO=0)\}$
- For P(NDO) : $\{P(NDO=0)\}$
- For P(DO|FO,NDO) : $\{P(DO=0|FO=0,NDO=0), P(DO=0|FO=0,NDO=1), P(DO=0|FO=1,NDO=0), P(DO=0|FO=1,NDO=1)\}$

Maximum Likelihood Estimation

Given a BN, a variable x_i, our database D, and a vector of values $w_{i,j}$ assigned to its vector of parent variables π_i, the MLE estimation for the conditional probability:

$$\theta_{i,j,k} = \Pr(x_i = k \mid \pi_i = w_{i,j})$$

is

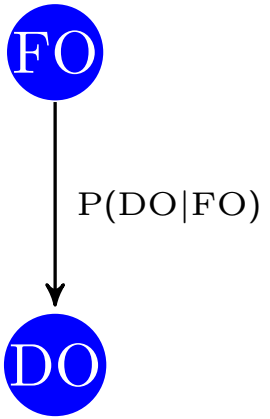
$$\hat{\theta_{i,j,k}} = \frac{N_{i,j,k}}{N_{i,j}}$$

assuming multinomial random variables, where:

- $N_{i,j,k} = |\{r \mid r \in D, r[x_i = k, \pi_i = w_{i,j}]\}|$. That is, the number of rows in D where the parent variables π_i have assigned the values $w_{i,j}$ and x_i the value k
- $N_{i,j} = \sum_{k=1}^{r_i} N_{i,j,k}$

Maximum Likelihood Estimation

Consider the following BN for the FO-DO problem and a database of cases D



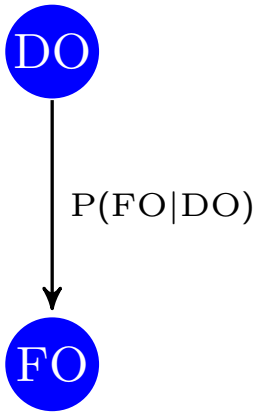
DO	FO
0	0
0	0
1	0
1	0
0	1
1	1
1	1
1	1

Estimations:

- $P(FO = 0) = \frac{4}{8}$
- $P(DO{=}0|FO{=}0) = \frac{2}{4}$, $P(DO{=}0|FO{=}1) = \frac{1}{4}$

Maximum Likelihood Estimation

Consider the alternative BN for the FO-DO problem and the same database of cases D



DO	FO
0	0
0	0
1	0
1	0
0	1
1	1
1	1
1	1

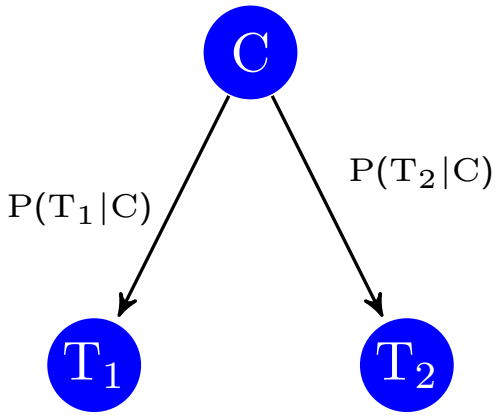
Estimations:

- $P(DO = 0) = \frac{3}{8}$
- $P(FO{=}0|DO{=}0) = \frac{2}{3}$, $P(FO{=}0|DO{=}1) = \frac{2}{5}$

What model predicts with less uncertainty the values for the variables ?

Maximum Likelihood Estimation

Consider the following BN for the two-tests cancer example



Such that we do not know its corresponding probability tables, but we have instead a database of samples, obtained from past clinical records

Maximum Likelihood Estimation

Suppose the following database D for the cancer problem:

C	T1	T2
0	0	1
0	0	0
0	0	0
0	1	0
0	0	0
0	0	0
0	0	0
1	1	1

We get the estimations:

- $P(C=0) = \frac{7}{8}$
- $P(T1=0|C=0) = \frac{6}{7}$, $P(T1=0|C=1) = \frac{0}{1}$
- $P(T2=0|C=0) = \frac{6}{7}$, $P(T2=0|C=1) = \frac{0}{1}$

Maximum Likelihood Estimation

Observe that depending on the particular database D, some probabilities may be estimated to be equal to 0, even if the real probability could be very small, but not 0 !

As this can produce problems when performing inference, some learning algorithms add an small bias factor so that probabilities equal to 0 are replaced by very small probabilities

Maximum Likelihood Estimation

For example, with the previous estimations for the two-tests cancer example, if we now want to compute:

$$P(C=1|T_1=1,T_2=0) = \frac{P(C=1)P(T_1=1|C=1)P(T_2=0|C=1)}{P(T_1=1,T_2=0)}$$

we get the following answer:

$$\frac{1}{8}\ \frac{1}{1}\ \frac{0}{1}\ \frac{1}{P(T_1=1,T_2=0)} = 0$$

So, because one of the factors was 0, independently of the fact that other factors were greater than 0, the final result is 0.

This crisp result is not very helpful !

Learning the Structure of a Bayesian Network

What is the best Bayesian Network that explains a given data set D ?

Ingredients for a learning algorithm for Bayesian Networks:

1. Scoring: When a Bayesian Network is better than other for explaining D ?
2. Search space: What is the search space of possible Bayesian Networks we want to consider ?
3. Search strategy: If the search space has exponential size, how do we search the best Bayesian Network ?

Scoring: How well a Bayesian Model Fits with our Data ?

Given our data set D, and a possible Bayesian Network BN for our problem, a possible scoring model for how good is BN is computing the probability of obtaining all the samples in D from BN:

$$P(D|BN)$$

The higher that value is, the more confident we can be with the model BN because the data set D will be closer to what BN tipically will generate when used as a model for simulating the outcomes obtained from the real problem

Scoring: How well a Bayesian Model Fits with our Data ?

Observe that from probability theory we have that:

$$P(D|BN)P(BN) = P(D, BN)$$

Typical scoring metrics developed for bayesian learning algorithms use rather $P(D, BN)$ than $P(D|BN)$, so that it is possible to include a prior probability $P(BN)$ on the space of possible bayesian network models

That is, to induce some bias towards some preferred bayesian models, if we have additional information that may indicate us that some models are more natural than others

Scoring: The Uniform Prior Scoring Model (UPSM)

We can compute $P(D, BN)$ integrating over all the possible probability distributions B_p consistent with BN:

$$P(D, BN) = P(BN) \int_{B_p} (P(D|BN, B_p) f(B_p|BN) dB_p)$$

Where:

- $P(D|BN, B_p)$ is the probability of observing the samples in D, assuming the data is generated with network BN and probability distribution B_p.
- $f(B_p|BN)$ is the probability of having probability distribution B_p assuming the correct network is BN, and assuming that all the B_p consistent with BN are uniformly distributed

Scoring: The Uniform Prior Scoring Model (UPSM)

In UPSM, the set of prior probabilities over the set of Bayes networks (BN) follows an uniform distribution (all the BN are equally likely in the absence of any data set D)

So, we can substitute P(BN) by an arbitrary constant C:

$$P(D, BN) = C \int_{B_p} P(D|BN, B_p) f(B_p|BN) dB_p$$

Scoring: The Uniform Prior Scoring Model (UPSM)

Assuming all the samples r_h from D are independently generated:

$$P(D|BN, B_p) = \prod_{h=1}^{|D|} P(r_h|BN, B_p)$$

Considering the factorization given by BN and B_p:

$$\prod_{h=1}^{|D|} P(r_h|BN, B_p) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} P(x_i = k|\pi_i = w_{i,j})^{N_{i,j,k}}$$

Finally, working further with the term $f(B_p|BN)$ we obtain:

$$P(D, BN) = C \prod_{i=1}^{n} \prod_{j=1}^{q_i} (r_i - 1)! \int_{\theta_{i,j,1}} \dots \int_{\theta_{i,j,r_i}} \prod_{k=1}^{r_i} \theta_{i,j,k}^{N_{i,j,k}} d\theta_{i,j,1} \dots d\theta_{i,j,r_i}$$

Scoring: The Uniform Prior Scoring Model (UPSM)

And working with Dirichlet's integrals:

$$P(D, BN) = C \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}!$$

This scoring scheme gives a higher value to a BN that predicts with higher probability the samples in D

Thats is, to a BN where the samples in D are more likely to be obtained.
Check the original paper where this scoring model (and the learning algorithm K2 that uses it) is presented !

Scoring: The Uniform Prior Scoring Model (UPSM)

Example: Consider the two possible models for the FO-DO problem and our previous database D

Estimations:

- $P(FO=0)=\frac{4}{8}$
- $P(DO=0|FO=0)=\frac{2}{4}$, $P(DO=0|FO=1)=\frac{1}{4}$
- UPSM:
  - For FO: $\frac{4!4!}{(8+1)!}$
  - For DO: $\frac{2!2!}{(4+1)!}\frac{3!1!}{(4+1)!}$
  - Final product: $0,000002646$

Estimations:

- $P(DO=0)=\frac{3}{8}$
- $P(FO=0|DO=0)=\frac{2}{3}$, $P(FO=0|DO=1)=\frac{2}{5}$
- UPSM:
  - For DO: $\frac{5!3!}{(8+1)!}$
  - For FO: $\frac{2!1!}{(3+1)!}\frac{2!3!}{(5+1)!}$
  - Final product: $0,000002756$

So, the model P(DO)P(FO|DO) seems to be slightly better (from the point of view of UPSM)

Scoring: The Uniform Prior Scoring Model (UPSM)

Example: Consider the two following models for the two-tests cancer problem and our previous database D

Factorization for BN_1:

$$P(C)P(T_1|C)P(T_2|C)$$

Factorization for BN_2:

$$P(T_1)P(C|T_1)P(T_2|C)$$

Factors for $P(BN_1, D)$:

	$j=0$	$j=1$
T_1	$\frac{6!1!}{(7+1)!}$	$\frac{0!1!}{(1+1)!}$
T_2	$\frac{6!1!}{(7+1)!}$	$\frac{0!1!}{(1+1)!}$
C	$\frac{7!1!}{(8+1)!}$	

Factors for $P(BN_2, D)$:

	$j=0$	$j=1$
T_1	$\frac{6!2!}{(8+1)!}$	
T_2	$\frac{6!1!}{(7+1)!}$	$\frac{0!1!}{(1+1)!}$
C	$\frac{6!0!}{(6+1)!}$	$\frac{1!1!}{(2+1)!}$

$P(BN_1, D) = \frac{1}{9 \cdot 8^3 \cdot 7^2 \cdot 4} = 11.07 \cdot 10^{-6}$ $P(BN_2, D) = \frac{1}{9 \cdot 8^2 \cdot 7^3 \cdot 6} = 8.44 \cdot 10^{-7}$

So, BN_1 seems to be a better model for explaining D with more accuracy

Scoring: The Uniform Prior Scoring Model (UPSM)

When has the factor associated with a variable x_i and parent row j in the score $P(BN, D)$:

$$\frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}!$$

a higher score ?

This factor is associated with the conditional probability:

$$P(x_i = k | \pi_i = w_{i,j})$$

Scoring: The Uniform Prior Scoring Model (UPSM)

Best case for UPSM: No uncertainty on x_i

There is a case v with $N_{i,j,v} = N_{i,j}$ (the other v′ terms are 0). Then:

$$\frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}! = \frac{(r_i - 1)! N_{i,j}!}{(N_{i,j} + r_i - 1)!}$$

That is, with parent case j the model predicts a unique possible value v for x_i (no uncertainty, perfect prediction!) Even better case: when

$N_{i,j,v} = N_{i,j}$ is much bigger than r_i ($r_i = o(N_{i,j})$)

Scoring: The Uniform Prior Scoring Model (UPSM)

Worst case for UPSM: Maximal uncertainty on x_i

All the $N_{i,j,k}$ are equal: $N_{i,j,k} = N_{i,j}/r_i$

$$\frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}! = \left(\frac{N_{i,j}}{r_i}\right)!^{r_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!}$$

That is, maximal uncertainty on the value predicted for x_i (all the possible values equally likely)

Even worse case: when $N_{i,j,k} = N_{i,j}/r_i = 1$ (so $N_{i,j} = r_i$). Then:

$$\left(\frac{N_{i,j}}{r_i}\right)!^{r_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} = \frac{(r_i - 1)!}{(r_i + r_i - 1)!}$$

Scoring: The Uniform Prior Scoring Model (UPSM)

Some relevant properties of the UPSM scoring model:

- Local scoring scheme: the factor contributed to the score by a node x_i depends only on its set of parents π_i, so it does not change when we change the parent set of other nodes

- Incremental computation: Due to the local scoring property, it can be computed incrementally for a BN that is incrementally build: this is helpful for the efficient implementation of greedy search algorithms

Scoring: Penalising Too Complex Models

Remember that in machine learning, a general principle is that simpler hypothesis are preferred when many of them seem to be equally good

Too complex hypothesis can overfit to the data: they will predict well our data set, but give poor answers with new data

With UPSM, adding more parent nodes to a node x_i will almost always increase its scoring, but it may lead to overfit as the model could capture particular dependencies of the data set D

Scoring: Penalising Too Complex Models

Remember that in machine learning, a general principle is that simpler hypotheses are preferred when many of them seem to be equally good

Too complex hypotheses can overfit to the data: they will predict well our data set, but give poor answers with new data

Scoring: Penalising Too Complex Models

A possible way to penalise too complex models is to substract from the UPSM scoring for a BN a measure of the size of the model:

$$\frac{\log |D|}{2} \sum_{i=1}^{n} q_i (r_i - 1)$$

where remember that q_i is the number of different cases associated with the parents π_i of x_i on the BN being scored

The scoring that incorporates the substraction of such model size is called Bayesian Information Criterion (BIC) or Minimum Description Length (MDL)

Looking For a Good Model

Once we have a way of scoring how good is a BN model with respect to our database D, the next step is to pick a search algorithm for learning the best possible BN for our problem, given our database D

In principle, the best option would be to perform systematic search trough the space of possible Bayesian networks that explain D

But even with the restriction that a BN is an acyclicic graph, the search space is still of exponential size !

So, the machine learning methods developed for Bayesian networks have focused on incomplete methods, based mainly on local search

A Greedy Local Search Algorithm: K2

How it works:

- Generate some ordering between the nodes
- Initialize the set of parents for every node to the empty set
- For each node x_i, incrementally and in a greedy fashion, selects its set of parent nodes π_i from the set of nodes that precede it in the ordering using UPSM to score each possible parent

Even if we work with a fixed ordering, still the possible number of Bayesian networks with n nodes is exponential:

$$2^{n(n-1)/2}$$

Exercise: Use Combinatorial Analysis to check why this is the number of networks for a fixed ordering between the nodes

A Greedy Local Search Algorithm: K2

Function LearnBNetK2(D ,Ordering, u)

for each v_i in Ordering do
  $\pi_i := \{\}$;
  $P_{old}(x_i) := UPSM(x_i, \pi_i, D)$;
  improving := true ;
  while improving $\wedge$ $|\pi_i| < u$ do
    Let z be a node $\in Prev(x_i, Ordering) \setminus \pi_i$ that maximizes
    $UPSM(x_i, \pi_i \cup \{z\}, D)\}$;
    $P_{new}(x_i) := UPSM(x_i, \pi_i \cup \{z\}, D)$;
    if $P_{new}(x_i) > P_{old}(x_i)$ then
      $P_{old}(x_i) := P_{new}(x_i)$;
      $\pi_i := \pi_i \cup \{z\}$;
    else
      improving := false ;

A Greedy Local Search Algorithm: K2

Using an upper bound on the number of parents for each node (u) on the algorithm is a way to overcome a possible overfit of the model obtained

But of course, it is not clear how to select at the beginning a good value for u

The local search scheme used by K2 can actually be used with any other scoring model that has the property of being local: the scoring given by a node x_i depends only on its set of parents π_i, and not on the parents of other nodes

Greedy Local Search is not Perfect

Database D:

x_i	p_1	p_2	p_3
1	0	0	1
0	0	0	0
1	0	1	0
0	0	1	1
0	1	1	1
1	1	1	0
0	1	0	0
1	1	0	1

Consider finding the best parent set for x_i with at most two nodes, taken from $\{p_1, p_2, p_3\}$
First, observe that:

$$\mathrm{UPSM}(x_i, \{\}, D) = \frac{4!4!}{9!} = \frac{1}{630}$$

But for any p_j:

$$\mathrm{UPSM}(x_i, \{p_j\}, D) = \frac{2!2!}{5!}\frac{2!2!}{5!} = \frac{1}{900}$$

Greedy Local Search is not Perfect

Database D:

x_i	p_1	p_2	p_3
1	0	0	1
0	0	0	0
1	0	1	0
0	0	1	1
0	1	1	1
1	1	1	0
0	1	0	0
1	1	0	1

Then, we have that starting with any p_i:

$$\mathrm{UPSM}(x_i, \{p_j\}, D) = \frac{2!2!}{5!}\frac{2!2!}{5!} = \frac{1}{900}$$

but the effect of adding a second parent depends on the first one:

- If we start with p_1, with $p_j = p_2$ or $p_j = p_3$:

$$\mathrm{UPSM}(x_i, \{p_1, p_j\}, D) = \left(\frac{1!1!}{3!}\right)^4 = \frac{1}{1296}$$

Greedy Local Search is not Perfect

Database D:

x_i	p_1	p_2	p_3
1	0	0	1
0	0	0	0
1	0	1	0
0	0	1	1
0	1	1	1
1	1	1	0
0	1	0	0
1	1	0	1

On the other hand, if we extend the maximum number of parents to three, starting with parent set $\{p_1\}$ the previous computations show that it makes no profit to add a second parent, but observe that:

$$\text{UPSM}(x_i, \{p_1, p_2, p_3\}, D) = \left(\frac{1!}{2!}\right)^8 = \frac{1}{256}$$

So, even if adding a second parent node to $\{p_1\}$ does not improve the score, adding it two more parents leads to a better score than only with $\{p_1\}$

But still observe that the best global solution is the parent set: $\{p_2, p_3\}$ that is never reached starting from $\{p_1\}$

Concluding: the greedy local search approach does not guarantee to find the best global set of parents !

The Speed Dating Problem

Speed Dating

From wikipedia: Speed dating is a formalised matchmaking process or dating system whose purpose is to encourage people to meet a large number of new people

How can we know when two persons are gonna match each other ?

Learning a Bayes Model for the Speed Dating Problem

Consider the following database of previous dates with a client, whith characteristics of the mates, and whether they were liked by our client

ideo	sex	talks	vege	blade	youlike
izq	mas	nada	si	si	si
izq	fem	nada	si	no	no
izq	fem	nada	si	si	si
cen	mas	poco	si	no	no
cen	mas	poco	no	si	si
cen	mas	poco	no	no	no
der	mas	mucho	no	si	no
izq	fem	mucho	no	no	si
der	fem	mucho	no	si	no
izq	fem	mucho	no	no	si

Is there any sound pattern with sex, blade and youlike ?

Learning a Bayes Model for the Speed Dating Problem

Can we learn a model to explain what makes people to be liked by our client ?

If we can learn this, we can try to reduce the number of unsuccessful meetings between our client and other people

Many dating companies on internet can take profit of such learning process

A possibility is to learn a bayesian model for the joint probability distribution of the variables of the problem

Learning a Bayes Model for the Speed Dating Problem

We run 10 times the K2 algorithm with a random ordering of its variables in each run and maximum number of parents 3. Results ordered by model score

Run #	Log(BayesScore)	Total # of edges
3	-51.52	7
10	-51.78	8
1	-52.22	8
5	-52.94	6
2	-54.31	9
9	-54.92	5
7	-55.11	6
8	-55.16	6
4	-56.32	6
6	-57.32	3

Learning a Bayes Model for the Speed Dating Problem

Suppose the following new person arrives

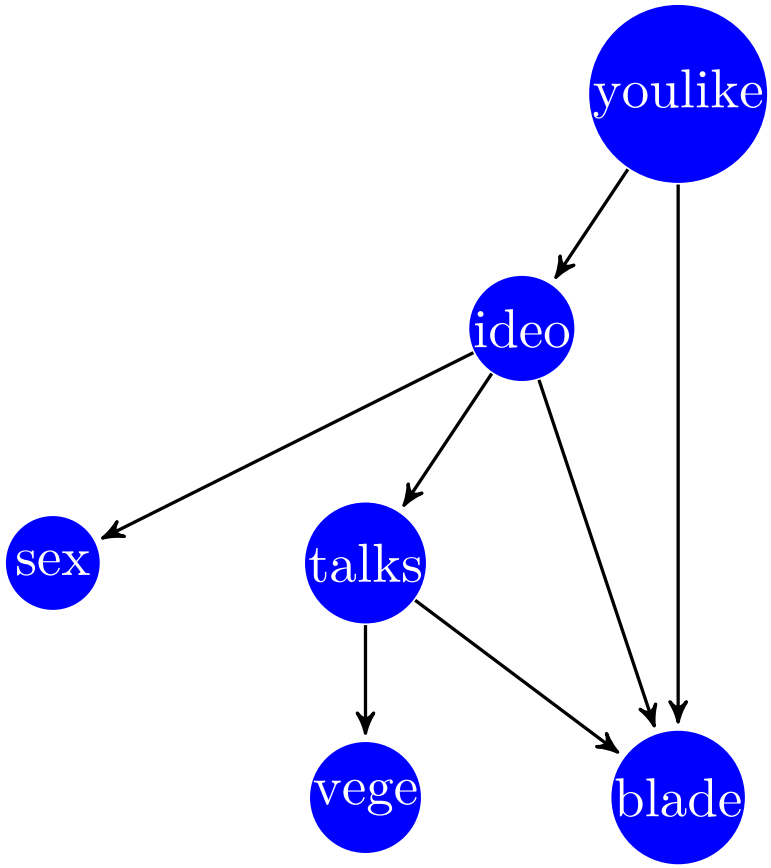
ideo	sex	talks	vege	blade	youlike
izquierdas	mas	nada	yes	no	yes

But if the person in new, we still do not know the value for the variable youlike. But we would like to predict it !

So, we would like to know how much likely are the results youlike=yes and youlike=no, given the known attributes of the new person (conditioned to them)

Learning a Bayes Model for the Speed Dating Problem

Bayesian network of best model found (# 3):



Learning a Bayes Model for the Speed Dating Problem

How well our ten learned models predict the value of the variable youlike for our test instance with known attribute values E ?

Run #	Log(BayesScore)	P(youlike = yes|E)
3	-51.52	0.156
10	-51.78	0.083
1	-52.22	0.083
5	-52.94	0.156
2	-54.31	0.083
9	-54.92	0.885
7	-55.11	0.67
8	-55.16	0.656
4	-56.32	0.67
6	-57.32	0.5

Learning a Bayes Model for the Speed Dating Problem

Observe that the models with the lowest bayes score (last 5 models of the previous table) are the ones that predict with high probability the right answer

In the data set used for learning, with these attributes the answer is almost always youlike=no

So, it is reasonable that a learning method focused on adjusting well to the training data (to explain it with high confidence) should give models where for these attributes the answer with highest probability is youlike=no

So, we have here a problem derived by the overfitting to the training data, that is a common problem with many other learning techniques