CES -161 - Modelos Probabilísticos em Grafos

Learning Probabilistic Models and

Knowledge Engineering with Bayesian Networks

Prof. Paulo André Castro
pauloac@ita.br
www.comp.ita.br/~pauloac
Sala 110, IEC-ITA

Learning Probabilistic Models

- Bayesian network structure may be constructed by:
  - hand, presumably during a process of eliciting causal and probabilistic dependencies from an expert or
  - learning via causal discovery or
  - a combination of both approaches
- However the structures are arrived at, they will be useless until parameterized, i.e., the conditional probability tables are specified, characterizing the direct dependency between a child and its parents.

Learning Probabilistic Models

- Learning probabilities
- Learning Models
- Bayesian Classifiers
- Evaluating Classifiers

Learning probabilities

- Parameterizing a categorial (or multinomial) model (variables with two or more possible values) can be done using a conjugate family of distributions, namely the Dirichlet family of distributions

- The Dirichlet distribution with τ possible values is written $D[\alpha_1, .., \alpha_i, .., \alpha_\tau]$ with α_i being the hyperparameter for value i and the probabilities of each value are given by:

$$P(X = i) = \frac{\alpha_i}{\sum_{j=1}^{\tau} \alpha_j}$$

Learning probabilities

- When we observe a data instance (or data point) where x=i then we can update the distribution from $D[\alpha_1,..,\alpha_i,..,\alpha_\tau]$ to $D[\alpha_1,..,\alpha_i+1,..,\alpha_\tau]$ and the probabilities can still calculated in the same way, but now it reflects the observed instances

$$P(X=i)=\frac{\alpha_i}{\sum_{j=1}^{\tau}\alpha_j}$$

- In fact when $\alpha_i = 1$ for all i, it is equivalent to the Laplace estimator (or smoothing).
- Note that small values of α_i make the data points more relevant and vice-versa

Multiple node

- The technique provides parameter estimation for BNs with a single node. What about multiple nodes?

- In order to parameterize an entire network, we can simply iterate over its nodes using a algorithm called Multinomial parameterization

- This algorithm is a very simple counting solution to the problem of parameterizing multinomial networks. This solution is certainly the most widely used and is available in the standard Bayesian network tools

Multinomial Parameterization

Algorithm *Multinomial Parameterization (Spiegelhalter and Lauritzen method)*

1. *For each node* X_j

   For each instantiation of Parents(X_j)*, assign some Dirichlet distribution for the* τ *states of* X_j $D[\alpha_1, \ldots, \alpha_i, \ldots, \alpha_\tau]$

2. *For each node* X_j

   For each joint observation of all variables $X_1, \ldots, X_k$

   (a) *Identify which state* i X_j *takes*

   (b) *Update* $D[\alpha_1, \ldots, \alpha_i, \ldots, \alpha_\tau]$ *to* $D[\alpha_1, \ldots, \alpha_i + 1, \ldots, \alpha_\tau]$ *for the distribution corresponding to the parent instantiation in the observation*

Learning Models (discrete causal structures)

- We saw how to parameterize a categorical(multinomial) causal structure with conditional probability tables, regardless of how the structure might have been found, i.e. We saw how to learn the CPT given a specific structure

- Now we will extend the picture of the machine learning of Bayesian networks by considering how to automate the learning of discrete causal structures

- However, we will look at learning of causal structure using Bayesian metrics, rather than orthodox statistical tests

- In other words, the algorithms here will search the causal model space $\{h_i\}$, with some metric aiming to select an h_i that maximizes the function

Problems

- So, there are two computationally difficult tasks these learners need to perform
- First, they need to compute their metric, scoring individual hypotheses. This scoring procedure may itself be computationally expensive
- Second, they need to search the space of causal structures, which, is known to be exponential
- Most of the search methods applied have been variants of greedy search

K2

- The first significant attempt at a Bayesian approach to learning discrete Bayesian networks without topological restrictions was made in 1991 (Cooper and Herskovits, 1991)
- Their approach is to compute the metric for individual hypotheses, $P(h_i|e)$ by brute force

- Since our goal is to find that h_i which maximizes P(hije), we can satisfy this by maximizing $P(h_i|e)$, as we can see from Bayes' theorem

$$\begin{aligned} P(h_i|e) &= \frac{P(e|h_i)P(h_i)}{P(e)} \\ &= \frac{P(h_i,e)}{P(e)} \\ &= \beta P(h_i,e) \end{aligned}$$

K2 - Simplifying assumptions

1. The data are joint samples and all variables are discrete
2. Samples are independently and identically distributed (i.i.d.).
3. The data contain no missing values. If, in fact, they do contain missing values, then they need to be filled in with some estimate method
4. For each variable Xk in hi and for each instantiation of its parents Parents(Xk), P(Xk = xjhi;q;p(Xk)) is uniformly distributed over possible values Xk = x. (where q is the parameter vector (e.g., conditional probabilities))
5. Assume the uniform prior over the causal model space; i.e. $P(h_i)=1/|\{h_i\}|$

Discussion of K2's assumptions

- In fact, the first three assumptions are used widely in machine learning community…
- Fourth assumption: if relevant prior knowledge is available, the uniformity assumption can be readily dismissed by employing non-uniform Dirchlet priors over the parameter space, as we also discussed there.
- Fifth assumption: The prior over causal models is uniform that crude though it may be, it is not likely to throw the search so far off that the best causal models are missed.

Given these simplyfing assumptions...

- Given these simplifying assumptions, Cooper and Herkovits (1991) showed that the joint probability can be given by:

$$P(h_i, e) = P(h_i) \prod_{k=1}^{N} \prod_{j=1}^{|\Phi_k|} \frac{(s_k - 1)!}{(S_{kj} + s_k - 1)!} \prod_{l=1}^{s_k} \alpha_{kjl}!$$

Where

- N is the number of variables.
- $|\Phi_k|$ is the number of assignments possible to $\pi(X_k)$.
- s_k is the number of assignments possible to X_k.
- α_{kjl} is the number of cases in sample where X_k takes its l-th value and $\pi(X_k)$ takes its j-th value.
- S_{kj} is the number of cases in the sample where $\pi(X_k)$ takes its j-th value (i.e., $\sum_{l=1}^{s_k} \alpha_{kjl}$).

K2

- The computation of Cooper and Herkovits's formula do $P(h_i, e)$ is polynomial, i.e., computing P given a particular h_i is tractable under the assumptions
- Ok, but the number of possible h_i is very big…Yes, it is!
- So, they made another simplifying assumption:
  - Assume we know the temporal ordering of the variables
- That way, the search space is greatly reduced. In fact, for any pair of variables either they are connected by an arc or they are not.

K2

- Limitation of the maximum number of parents are also commonly required, as a way to reduce the search space

- Many variations can be used with K2, for instance change the metric from Bayesian to others metrics as: Minimum Description Length (MDL), Akaike Information Criterion (AIC), Entropy and others

- Machine learning frameworks provide several alternative methods for building Bayes networks, K2 is one the most relevant methods

Diabetes example

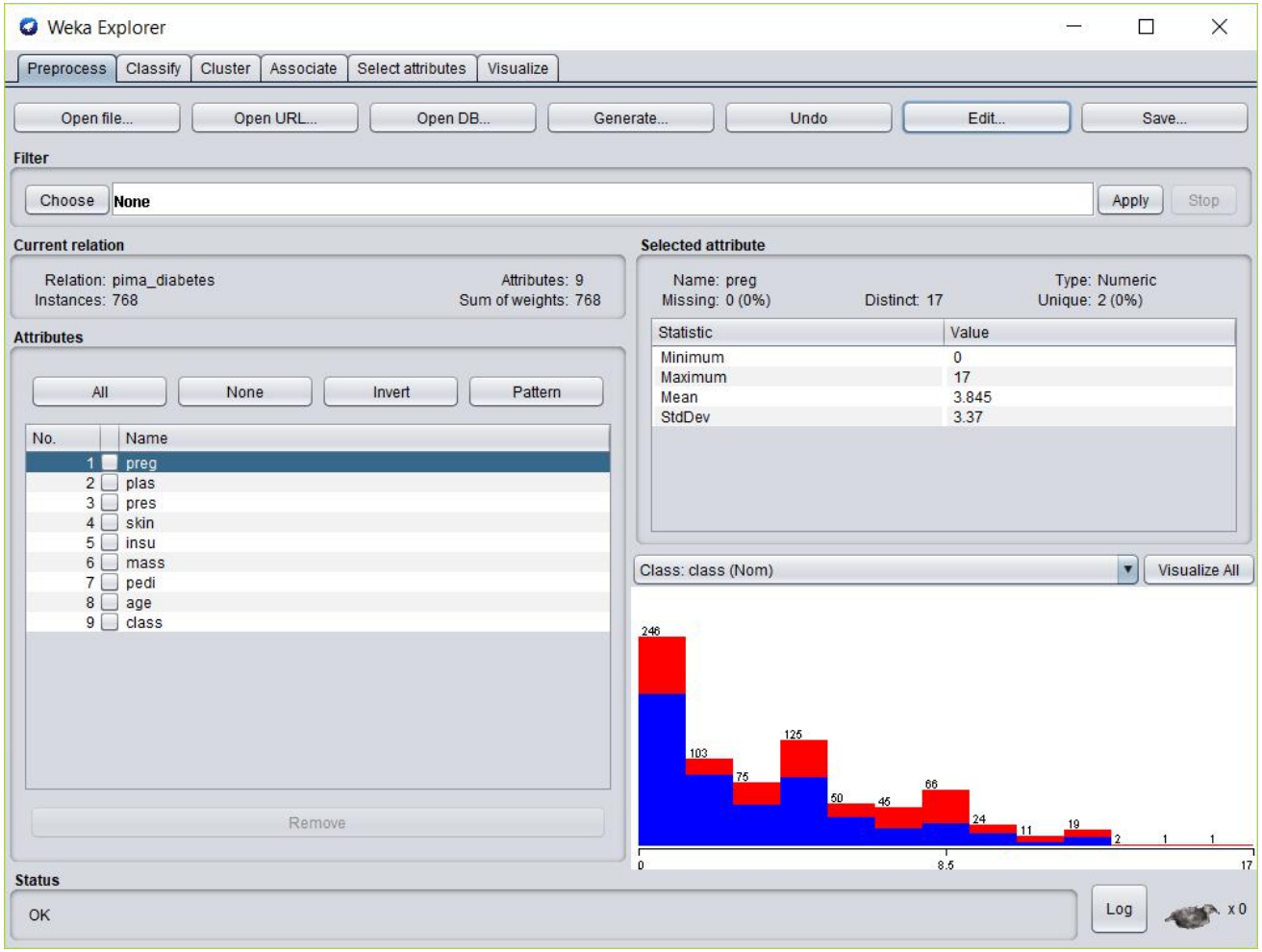
- The diagnostic, (binary-valued variable) investigated is whether the patient shows signs of diabetes according to World Health Organization criteria

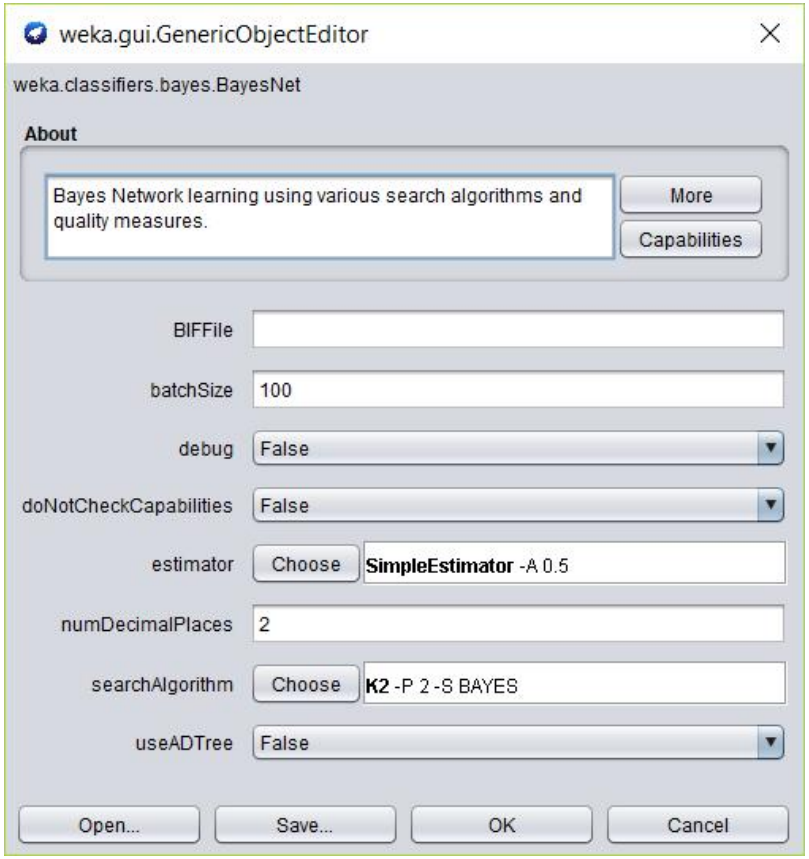
- Dataset: 768 instances, all patients **are females at 21 years old or more** of Pima Indian heritage. (Smith,J.W., Everhart, J.E., Dickson, W.C., Knowler,W.C., Johannes,R.S. ,1988)

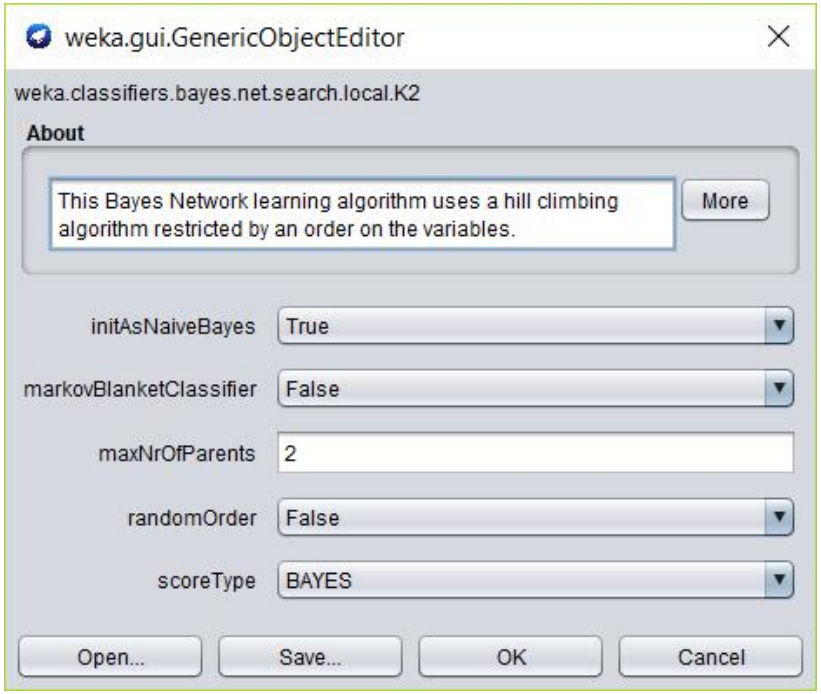
Diabetes example

- Variables:
  1. Number of times pregnant
  2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
  3. Diastolic blood pressure (mm Hg)
  4. Triceps skin fold thickness (mm)
  5. 2-Hour serum insulin (mu U/ml)
  6. Body mass index (weight in kg/(height in m)^2)
  7. Diabetes pedigree function
  8. Age (years)
  9. Class variable (class value 1 is interpreted as "tested positive for diabetes)

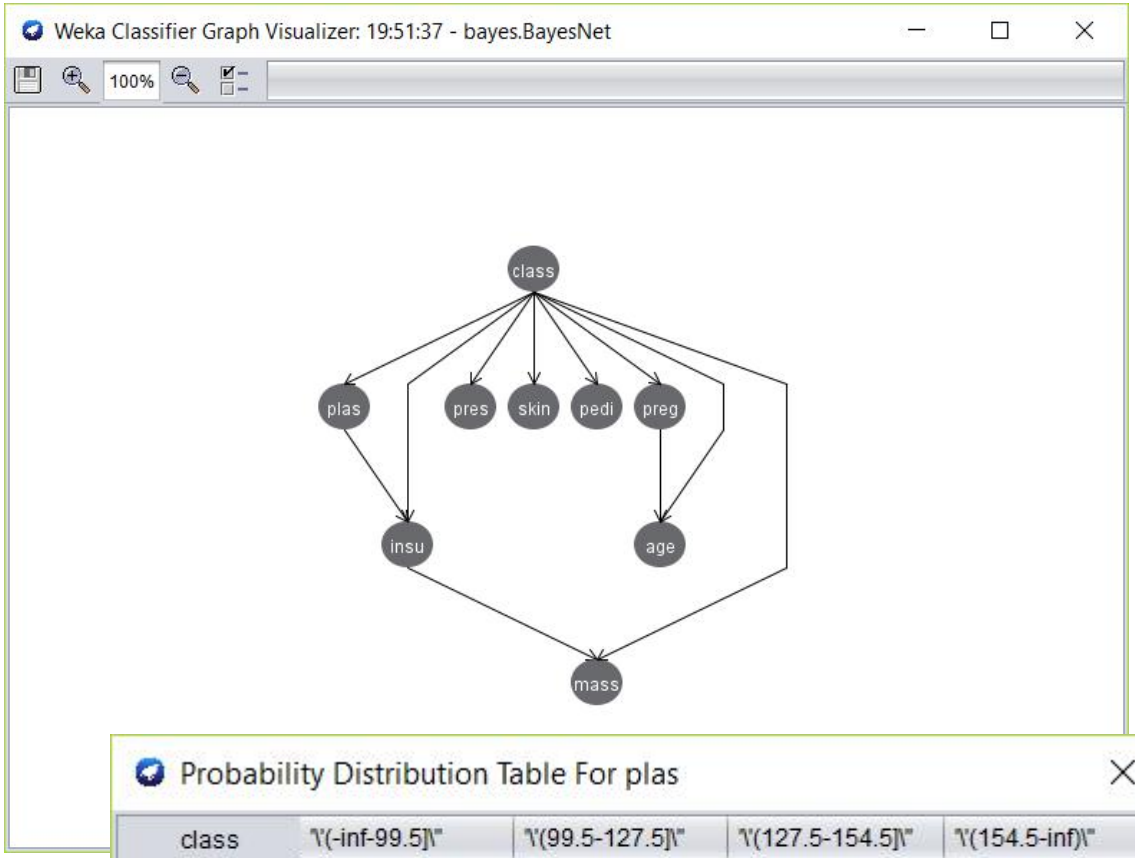
One simple tool (Weka)




weka.gui.GenericObjectEditor
weka.classifiers.bayes.BayesNet
About
Bayes Network learning using various search algorithms and quality measures.
More
Capabilities
BIFFile
batchSize
100
debug
False
doNotCheckCapabilities
False
estimator
Choose
SimpleEstimator -A 0.5
numDecimalPlaces
2
searchAlgorithm
Choose
K2 -P 2 -S BAYES
useADTree
False
Open...
Save...
OK
Cancel


weka.gui.GenericObjectEditor
weka.classifiers.bayes.net.search.local.K2
About
This Bayes Network learning algorithm uses a hill climbing algorithm restricted by an order on the variables.
More
initAsNaiveBayes
True
markovBlanketClassifier
False
maxNrOfParents
2
randomOrder
False
scoreType
BAYES
Open...
Save...
OK
Cancel

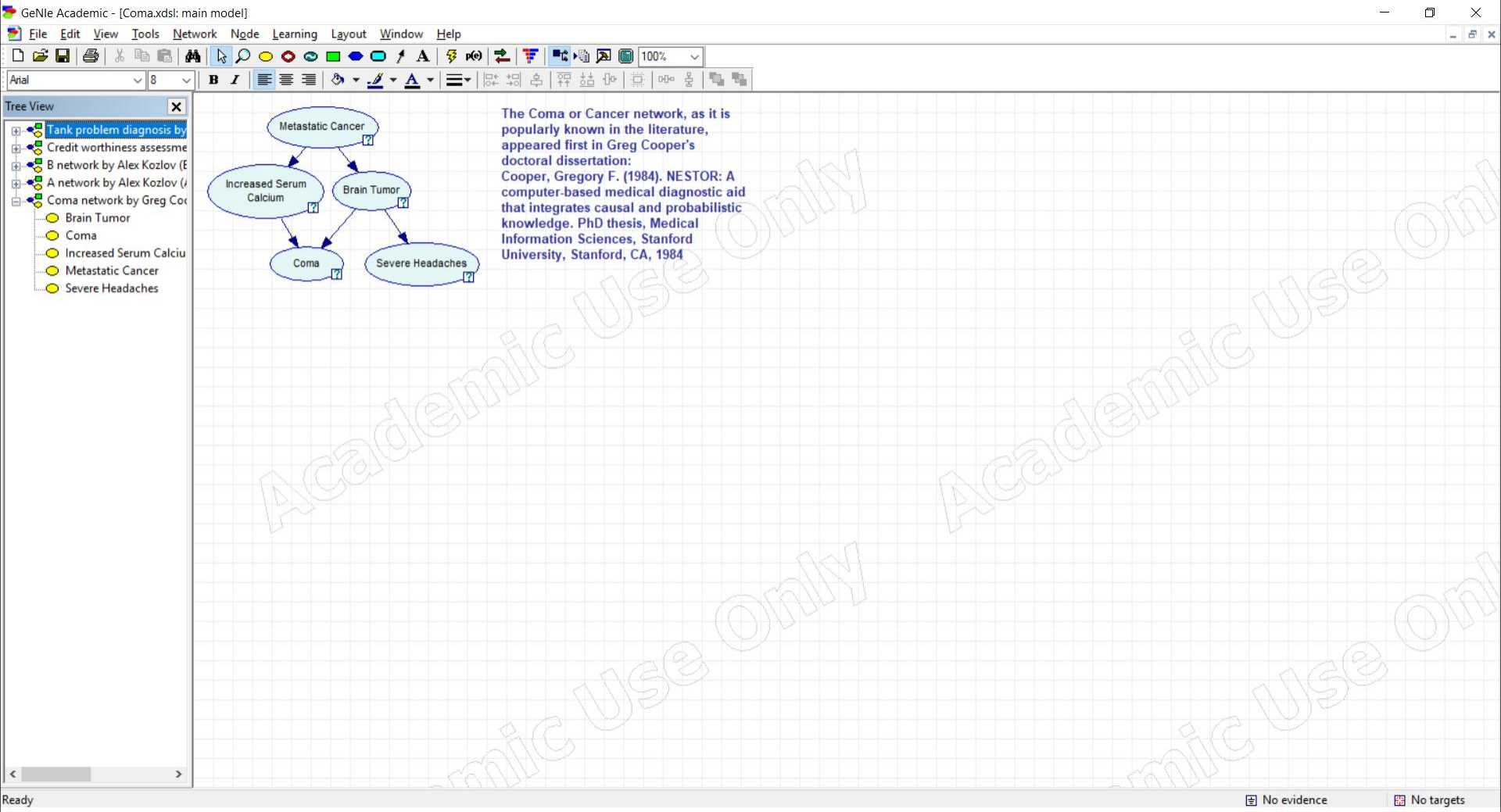
BN built by the dataset using K2, number of parents equal to 2 and estimator parameter equals to 0.5



Probability Distribution Table For plas

class	'\'(-inf-99.5]\''	'\'(99.5-127.5]\''	'\'(127.5-154.5]\''	'\'(154.5-inf)\''
tested_negative	0.362	0.419	0.17	0.049
tested_positive	0.061	0.291	0.283	0.365

Another tool: GeNie Academic



Other tools (Korb, Nicholson, 2011)

Name	Authors	Src	API	Exec	GUI	D/C	DN	DBN	OOBN	Params	Struct	D/U	Infer	Free
Ace	Darwiche (UCLA)	N	Y	WUM	N	D	N	N	N	N	N	All?	AC	
AgenaRisk	Agena	N	Y	WU	Y	Cd	N	?	Y	Y	Y	D	JT	$
AISpace (CISpace)	Poole et al. (UBC)	J	N	WU	Y	D	Y			N	N	D	VE	0
Analytica	Lumina	N	Y	W	Y	G, Cs, Cd	Y	Y	N	N	N	D	IS,G	$
B-course	U. Helsinki	N	N	WUM	Y	Cd	N	N	N	Y	Y	D	?	0
Banjo	Hartemink	J	Y	WUM	N	Cd	N	Y	N	N	Y	D	N	0
Bassist	U. Helsinki	C++	Y	U	N	G	N	?	?	Y	N	D	MH	0
BayesBuilder	Nijman (U. Nijmegen)	N	J	WUM	Y	D	N		N	N	N	D	JT	0
BayesiaLab	Bayesia Ltd	N	Y	WUM	Y	Cd	N	Y	?	Y	Y	CG	JT,G	$
Bayesware Discoverer	Bayesware	N	N	WUM	Y	Cd	N			Y	Y	D	?	$
bn4r	Bel, Dahl	Ru	Y	WUM	N	D	N	N	N	N	N	D	?	0
BNT	Murphy (U.C.Berkeley)	M/C	Y	WUM	N	G	Y	Y	N	Y	Y	UD	++	0
BucketElim	Rish (U.C.Irvine)	C++	Y	WU	N	D	N	N	N	N	N	D	VE	0
BUGS	MRC/Imperial College	N	N	WU	Y	Cs	N		N	Y	N	D	GS	0
CaMML	Wallace,Korb (Monash)	N	N	WU	Y	D	N	N	N	Y	Y	D	N	0
Causeway/Siam	SAIC	N	N	W	Y	?	Y	?	Y	?	?	D	0	$
DBL Interacive	Smith (U.Qld)	N	N	I	Y	D	Y	N	N	?	?	D	?	$
DBNbox	Roberts et al	M	-	-	N	Y	N			Y	N	D	++	0
Dezide Advisor	Dezide	N	Y	W	Y	N	Y		S	Y	N	?	PT	
dlib	King, Davis	C++	Y	WUM	Y	D	N	N	N	N	N	D	JT,G	
Elvira	Collaboration	J	Y	WUM	Y	Cx	Y		N	Y	Y	U	JT,VE,IS	
GDAGsim	Wilkinson (U. Newcastle)	C	Y	WUM	N	G	N			N	N	D	E	0
GeNIe/SMILE	U. Pittsburgh	N	WU	WU	Y	D	Y	Y	S	N	N	D	JT	0

Other tools - 2

Name	Authors	Src	API	Exec	GUI	D/C	DN	DBN	OOBN	Params	Struct	D/U	Infer	Free
GMRFlib	Rue, H	C/F	Y	WUM	N	G	N	N	N	Y	Y	U	JT	
GMTk	Bilmes (UW), Zweig (IBM)	N	Y	U	N	D	N			Y	Y	D	JT	0
Grappa	Green (Bristol)	R	-	-	N	D	N			N	N	D	JT	0
Hugin Expert	Hugin	N	Y	WUM	Y	G,Cd	Y	Y	Y	Y	CI	D,CG	JT	$
Hydra	Warnes (U.Wash.)	J	-	-	Y	Cs	N			Y	N	UD	MC	0
Java Bayes	Cozman (CMU)	J	Y	WUM	Y	D	N	N	N	N	N	D	VE	0
LibB	Friedman (Hebrew U)	N	Y	W	N	D	N			Y	Y	D	N	0
MASTINO	Mascherini	R	Y	WUM	N	G	N	N	N	Y	Y	D	SL	0
MIM	HyperGraph Software	P	Y	W	Y	G	N	N	N	Y	Y	UD,CG	JT	$
MSBNx	Microsoft	N	Y	W	Y	D	Y			N	N	D	JT	0
Netica	Norsys	N	Y	WUM	Y	Cs,Cd	Y	Y	N	Y	Y	D	JT,S	$
Optimal Reinsertion	Moore, Wong (CMU)	N	N	WU	N	D	N			Y	Y	D	N	0
PMT	Pavlovic (BU,Rutgers)	M/C	-	-	N	D	N			Y	N	D	O	0
PNL	Eruhimov (Intel)	C++	-	-	N	D	N			Y	Y	UD	JT	0
ProBT	Probayes	N	Y	WUM	N	Cs,Cd	Y		D	Y	Y	D	++	
Pulcinella	IRIDIA	L	Y	WUM	Y	D	N			N	N	D	?	0
Quiddity	IET	N	Y	Y	T	?	Y	?	?	?	?	D	?	$
RISO	Dodier (U.Colorado)	J	Y	WUM	Y	G	N	N	N	N	N	D	PT	0
Sam Iam	Darwiche (UCLA)	N	N ?	WU	Y	G ?	Y	N	N	Y?	N	D	CO	0
Tetrad IV	CMU	N	N	WU	Y	G,Cx	N	N	N	Y	CI	UD	N	0
UnBBayes	Ladeira, Carvalho	J	Y	Y	Y	D	Y	Y	Y	Y	Y	D	++	0
Vibes /Infer.NET	Winn & Bishop (Cambridge)	C#	Y	WU	Y	Cx	N		N	Y	N	FG	++	0
WinMine	Microsoft	N	N	W	Y	Cx	N			Y	Y	UD	N	0

Table explanation

Src Is the source code included? **N=no**. If yes, what language? **J** = Java, **M** = Matlab, **L** = Lisp, **C**, **C++**, **C#**, **R**, **A** = APL, **P** = Pascal, **Ru** = Ruby, **F** = Fortran.

API Is an application program interface included?

N means the program cannot be integrated into your code, i.e., it must be run as a standalone executable. **Y** means it can be integrated.

Exec The **executable** runs on: **W** = Windows (95/98/2000/NT), **U** = Unix, **M** = Mac-Intosh, **-** = Any machine with a compiler, or Java Virtual machine.

GUI Is a **G**raphical **U**ser **I**nterface included? **Y**=Yes,**N**=No.

D/C Are continuous-valued nodes supported (as well as discrete)? **G** = (condition-ally) Gaussians nodes supported analytically, **Cs** = continuous nodes supported by sampling, **Cd** = continuous nodes supported by discretization, **Cx** = contin-uous nodes supported by some unspecified method, **D** = only discrete nodes supported.

DN Are decision networks/influence diagrams supported? **Y**=Yes,**N**=No.

DBN Are dynamic Bayesian networks/influence diagrams supported? **Y**=Yes,**N**=No, **T** means some modeling over time but not with DBNs.

OOBN Are Object-oriented Bayesian networks/influence diagrams supported? **Y**=Yes,**N**=No, **S**=Not full OO, but support for subnetworks.

Params Does the software functionality include parameter learning? **Y**=Yes,**N**=No.

Struct Does the software functionality include structure learning? **Y**=Yes,**N**=No.

IC means **Y**, using conditional independency tests

K2 means **Y**, using Cooper & Herskovits' K2 algorithm

Table explanation

D/U What kind of graphs are supported? **U** = only undirected graphs, **D** = only directed graphs, **UD** = both undirected and directed, **CG** = chain graphs (mixed directed/undirected).

Inf Which inference algorithm is used?

JT = Junction Tree, **VE = variable (bucket) elimination**, **CO = conditioning**, **PT = Pearl's polytree**, **E = Exact inference** (unspecified), **MH = Metropolis Hastings**, **MC = Markov chain Monte Carlo (MCMC)**, **GS = Gibbs sampling**, **IS = Importance sampling**, **S = Sampling**, **O = Other** (usually special purpose), **++ = Many** methods provided, **? =** Not specified, **N =** None, the program is only designed for structure learning from completely observed data.

NB: Some packages support a form of sampling (e.g., **likelihood weighting**, **MDMC**), in addition to their exact algorithm; this is indicated by **(+S)**.

Free Is a free version available? **O=**Free (though possibly only for academic use), **$ =** Commercial (although most have free versions which are restricted in various ways, e.g., the model size is limited, or models cannot be saved, or there is no API.)

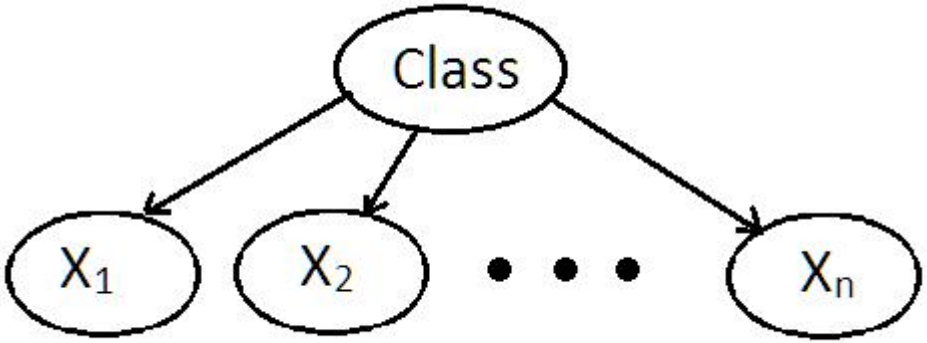
Bayesian Classifiers

- We have seen the use of Bayesian Network as classifiers. This kind of use has a substantial and growing body of work.
  - We mean without any expectation or interest in them as causal, explanatory models

- The basic approach is Naïve Bayes, but it can be extended in some simple ways as TAN

Naïve Bayes

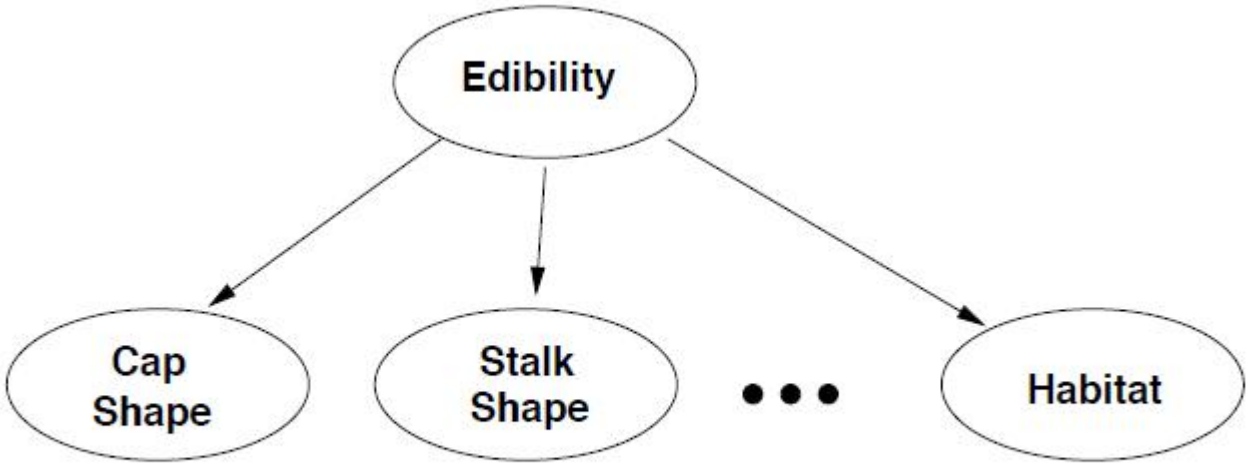
- We have already seen Naïve Bayes classifiers, which are in some sense the opposite of causal models but many times can be really good classifiers



Tree Augmented Naive Bayes (TAN)

- Tree Augmented Naive Bayes (TAN) models are perhaps the first natural step in desimplifying naive Bayes models

- For example, the mushroom problem from the UCI machine learning archive the target class variable is binary, indicating whether or not a mushroom is edible; there are 22 other variables describing attributes, such as color and shape, which may or may not be predictive of edibility

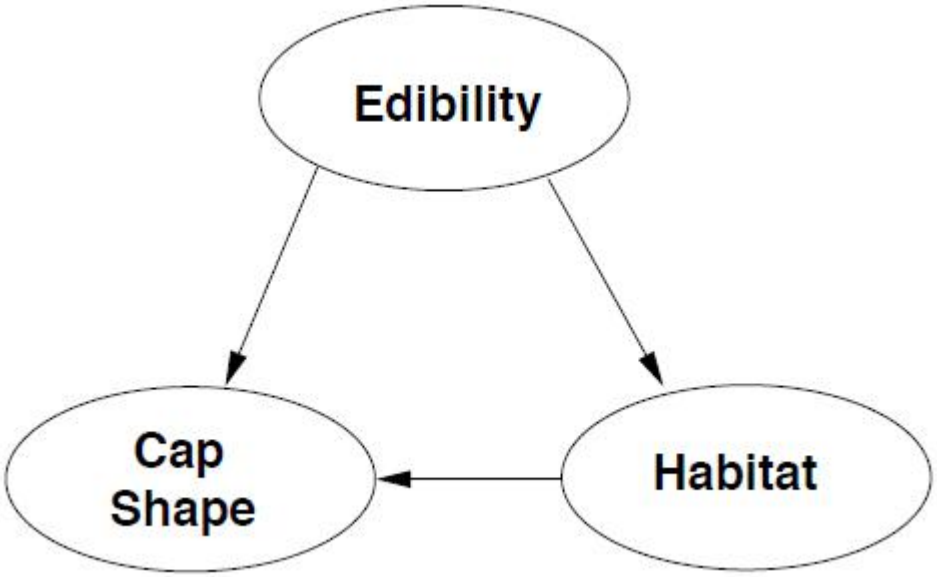


A naive Bayes mushroom model.

Tree Augmented Naive Bayes (TAN)

- TANs relax the Independence assumptions of NB by allowing some arcs directly between attributes, in particular allowing a tree amongst the attributes to be constructed, separately from their direct relations with the class variable

- For example, below it is a (very simple) TAN for the mushroom case, restricted for simplicity to two attributes



Some Results

- Friedman et al. (1997) performed empirical studies comparing NB, TAN and C4.5 (decision tree algorithm), along with other methods, using many of the UC Irvine machine learning data sets (Murphy and Aha, 1995)

- Naïve Bayes and C4.5 performed about the same, while TAN **almost dominated both**, meaning TAN performed either the about same or, in some cases, better, as measured by predictive accuracy

NB, TAN vs Causal Models

- Friedman et al. (1997) also compared the predictive accuracy of NB and TAN with full causal discovery algorithms
- They found NB and TAN **outperformed causal discovery** when there were a large number of attributes in the data.
- Causal discovery can potentially return any Bayesian network containing the attribute and class variables.
- If it happens to find the true model, the model which actually generated the data available, then that model (disregarding any noise in the data) **will necessarily be the best predictor** for the target variable.

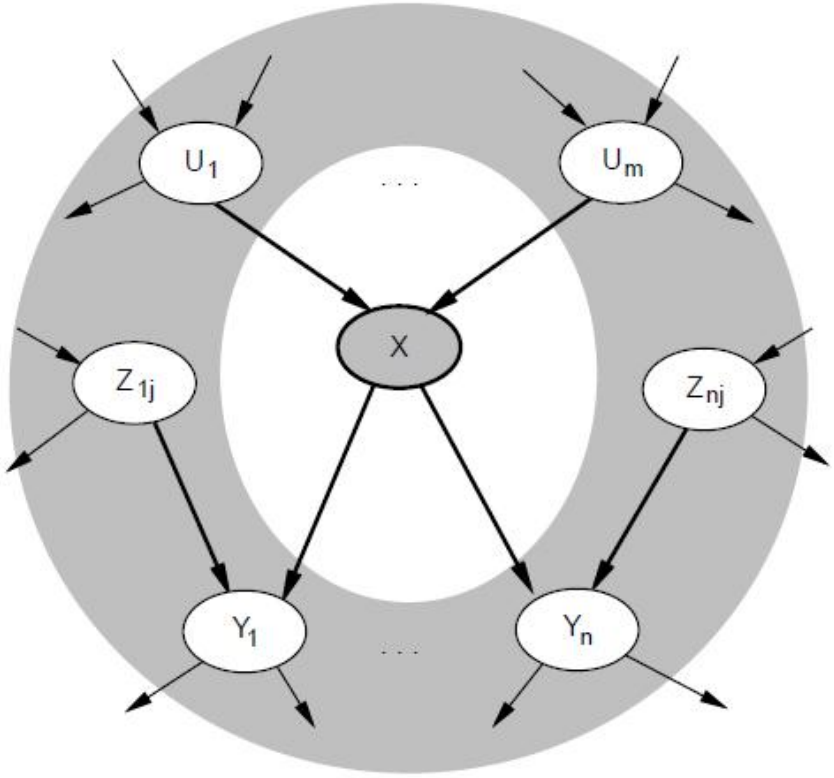
Problem with full causal

- The problem with full causal discovery for prediction, however, is the same as the problem with any kind of feature selection or model selection:
  - frequently the true model is not what is learned, but some similar, yet different, model is learned instead.

- The result may be that variables that should be in the target variable's Markov blanket are not and so are ignored during prediction, with potentially disastrous consequences for predictive accuracy.

Markov Blanket (Cobertor de Markov)

Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



Problems with NB and TAN

- NB and TAN, on the contrary, include all the attributes in their predictions, so this source of error is not even possible.

- To be sure, by being all-inclusive NB and TAN introduce a different potential source of error, namely **overfitting**.

- It may be that some variables are directly associated with the target variable only accidentally, due to noise in the available data

Problems with NB and TAN

- To compensate for this one may introduce **variable selection**, eliminating those attributes from the model which are contributing little to the prediction.

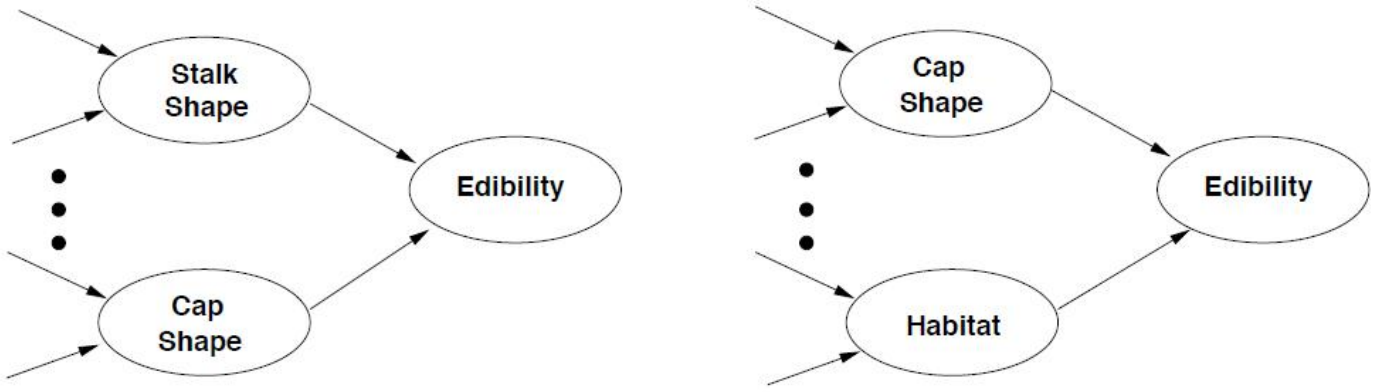
- In the latter case, again, incorrect variable selection returns us to the problem faced by causal discovery for prediction: variables missing from the target's Markov blanket

Ensemble Bayes models

- Another response to the problem of incorrectly identifying the Markov blanket, aside from utilizing all attributes as predictors, is to move to ensembles of predictive models.

- This means mixing the predictions of some number of distinct models together, using some weighting over the models

- For example, (below) there are two alternative (partial) Bayesian networks for the mushroom problem (note that they are not NB models)



Two BN models explaining (and predicting) edibility.

Evaluating Classifiers

- There is no agreed standard for how to assess the performance of classifiers.
- There are some common practices, but many of them are unjustified, or even unjustifiable in some domain
- Here we introduce the most commonly used approaches, together with their strengths and weaknesses, and some possible improvements upon them

Confusion Matrix

- Binomial target (class)

		Predicted class: yes	Predicted class: no
Actual class	yes	true positive	false negative
	no	false positive	true negative

- Multinomial target (class)

		Predicted class: a	b	c	Total
Actual class	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	Total	120	60	20	

Evaluation of Classifiers

- "Não é difícil criar classificadores, mas pode ser muito difícil criar bons classificadores…"
- Predictive accuracy or success rate (Taxa de acerto)
- Predictive accuracy is far and away the most popular technique for evaluating predictive models

  - Tx. Ac=(TP+TN)/(TP+TN+FP+FN) ou
  - Tx. Ac= ∑ (Elem da diagonal principal) / ∑ (Elem da matriz de confusão)

- É sempre uma boa medida?
  - Considere um sistema de detecção de fraude em transações de cartão de crédito que sempre diz que não há fraude….A taxa de acerto seria alta ou baixa? É um bom classificador?

Evaluation of classifiers

- Many domains deals with the problem of evaluating classifiers
- Predicted Accuracy (Success rate) is widely used, but there are other measures (sometimes) used with different names in different domains
- In information retrieval, a Web search engine produces a list of hits that represent documents supposedly relevant to the query.

Information retrieval – other measures

- Compare one system that locates 100 documents, 40 of which are relevant, with another that locates 400 documents, 80 of which are relevant. Which is better?

- The answer should now be obvious: it depends on the relative cost of false positives, documents that are returned that aren't relevant, and false negatives, documents that are relevant that aren't returned

- Information retrieval researchers define parameters called *recall* and *precision:*

$$\text{recall} = \frac{\text{number of documents retrieved that are relevant}}{\text{total number of documents that are relevant}}$$

$$\text{precision} = \frac{\text{number of documents retrieved that are relevant}}{\text{total number of documents that are retrieved}}$$

Other measures...

- ***precision***=TP/(TP+FP)

- ***recall*** =TP/(TP+FN) (also called *tp rate*)

- In different context, one measure can have other names. In medicine, the term sensitivity is used rather than recall.
    - **Sensitivity**: proportion of people with disease who have a positive test result: (TP / (TP + FN))
    - **FP rate**: proportion of people without disease who have a positive test result: (FP/ (FP+TN)). For the FP rate, small numbers are better

Other measures 2...

- **Specificity**: proportion of people without disease who have a negative test result: (TN/ (FP + TN)). It is equal to (1-**FP rate**)

- Sometimes the product *sensitivity* × specificity is used as an overall measure

- *Also used* ***F-measure***

$$\frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

- And the old success rate : (TP+TN)/(TP+FP+TN+FN)

Análise de Classificadores -2

- O problema com a taxa de acerto (e outras..) é que não levam em consideração os acertos por puro acaso…
- Uma alternativa: estatística kappa (Cohen's kappa)
  - $\kappa = (p_o - p_e)/(1-p_e)$
  - p_o é a concordância observada
  - p_e é a concordância esperada
- Kappa mensura o ganho em relação a distribuição esperada aleatória, 0 significa que não faz melhor do que ela e 1 significa perfeita acurácia.
- Um problema da estatística kappa (e também da taxa de acerto) é que não leva em consideração o custo dos erros…que podem ser diferentes e bem mais significativos que outros
- Cada parâmetro de comparação é parcial e deve-se fazer uma análise vários parâmetros (há vários outros...precision, f-measure,etc) considerando as particularidades do domínio do problema

Exemplo

- O classificador A prevê uma distribuição de classes com <0.6;0.3;0.1>, se o classificador fosse independente da classe real haveria uma distribuição na mesma proporção (Expected distribution)

		Predicted class			
		a	b	c	Total
Actual class	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	Total	120	60	20	

(a) Classifier A

		Predicted class			
		a	b	c	Total
Actual class	a	60	30	10	100
	b	36	18	6	60
	c	24	12	4	40
	Total	120	60	20	
		60%	30%	10%	

(b) Expected Distribution

- p_o = 88+ 40+12 = 140/200 (concordância observada)
- p_e = 60+18+4= 82/200 (concordância esperada)
- $\kappa = (p_o - p_e)/(1-p_e)$ = (140-82)/ (200-82) = 52 / 118 = 49,2%

Kappa statistic Interpretation

- *Cohen's Kappa*: Measures relative improvement on random predictor: 1 means perfect accuracy, 0 means we are doing no better than random

Value of Kappa	Level of Agreement
0-.20	None
.21-.39	Minimal
.40-.59	Weak
.60-.79	Moderate
.80-.90	Strong
Above .90	Almost Perfect

- (McHugh, 2012) Interrater reliability: the kappa statistic. Marry L. McHugh.

Cost of errors

Default cost matrixes: (a) a two-class case and (b) a three-class case.

		Predicted class	
		yes	no
Actual class	yes	0	1
	no	1	0

(a)

		Predicted class		
		a	b	c
Actual class	a	0	1	1
	b	1	0	1
	c	1	1	0

(b)

When error is not uniform?

- Problem: Predicting return of financial investment (very low, low, neutral, high, very high). Is it uniform?

$$\begin{bmatrix} L2 & L1 & N & H1 & H2 & \\ 0 & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & L2 \\ c_{2,1} & 0 & c_{2,3} & c_{2,4} & c_{2,5} & L1 \\ c_{3,1} & c_{3,2} & 0 & c_{3,4} & c_{3,5} & N \\ c_{4,1} & c_{4,2} & c_{4,3} & 0 & c_{4,5} & H1 \\ c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & 0 & H2 \end{bmatrix}$$

Cost-sensitive classification

- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally, we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Choose column (class) that minimizes expected cost
- This is the minimum-expected cost approach to cost-sensitive classification

Cost-sensitive learning

- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs

- Decision Networks can take cost into account using utility function…

Summary

- We have seen algorithm to learn probabilities (CPT, conditional probabilities tables) and also to learn Bayes network strucutures from datasets

- We have seen some Bayesian Classifiers and criteria to evaluate classifiers (Bayesian or non-Bayesian classifiers)
  - The process of building classifiers quite often uses the implicit i.i.d hyphotesis. In the next classes, we are going to discuss some challenging aspects of Machine learning when applied to Financial environments

- Now, let's discuss deeply the process of creating Bayesian Network for real-world scenarios

CES -161 - Modelos Probabilísticos em Grafos

Knowledge Engineering with Bayesian Networks

Prof. Paulo André Castro
pauloac@ita.br
www.comp.ita.br/~pauloac
Sala 110,
IEC-ITA

Knowledge Engineering with Bayesian Networks

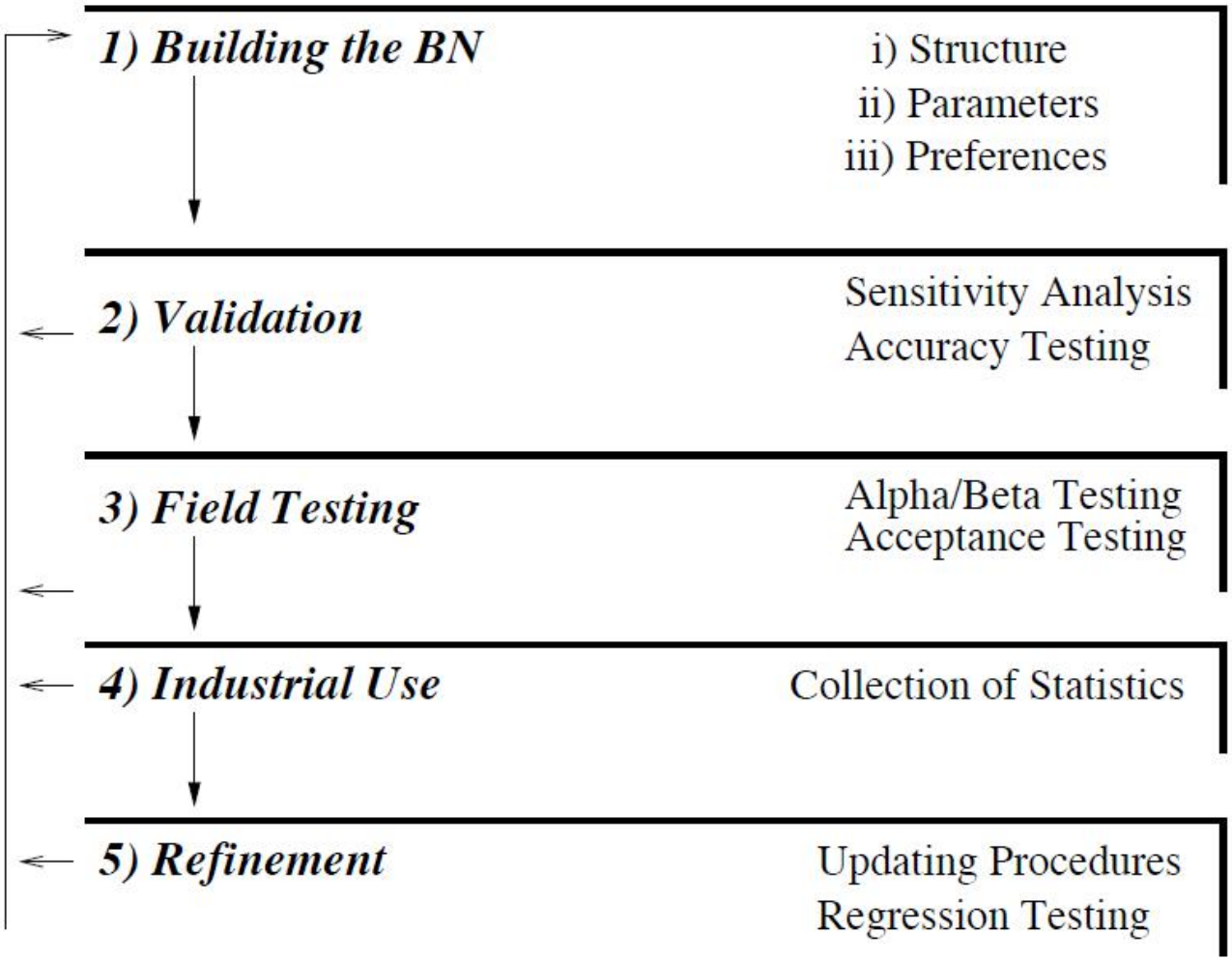
- When constructing a Bayesian network, the major modeling issues that arise are:
  - What are the variables? What are their values/states?
  - What is the graph structure?
  - What are the parameters (probabilities)?
- When building decision nets, the additional questions are:
  - What are the available actions/decisions, and what impact do they have?

KEBN life cycle model

- A simple view of the software engineering process construes it as having a lifecycle: the software is born (design), matures (coding), has a lengthy middle age (maintenance) and dies of old age (obsolescence).

- One effort at construing KEBN in such a lifecycle model (also called a “waterfall” model) is shown next.

KEBN "waterfall" life cycle model



Phases

- In the **building phase**, the major network components of structure, parameters and, if a decision network, utilities (preferences) must be determined through elicitation from experts, or learned with data mining methods, or some combination of the two

Phases - Validation

- **Validation** aims to establish that the network is right for the job, answering such questions as:
  - Is the predictive accuracy for a query node satisfactory?
  - Does it respect any known temporal order of the variables?
  - Does it incorporate known causal structure?

- **Sensitivity analysis** looks at how sensitive the network is to changes in input and parameter values, which can be useful both for validating that the network is correct and for understanding how best to use the network in the field

Phases – Field Testing

- **Field testing** first puts the BN into actual use, allowing its usability and performance to be gauged.
  - **Alpha testing** refers to an intermediate test of the system by inhouse people who were not directly involved in developing it; for example, by other inhouse BN experts.
  - **Beta testing** is testing in an actual application by a "friendly" end-user, who is prepared to accept hitting bugs in early release software.
  - **Acceptance testing** is surely required: it means getting the end users to accept that the BN software meets their criteria for use.

Phases – Industrial Use and Refinement

- **Industrial use** sees the BN in regular use in the field and requires that procedures be put in place for this continued use.
  - It is a good idea to collect statistics on the performance of the BN and statistics monitoring the application domain, in order to further validate and refine the network.

- **Refinement** requires some kind of change management regime to deal with requests for updating or fixing bugs. **Regression testing** verifies that any changes do not cause a degradation in prior performance

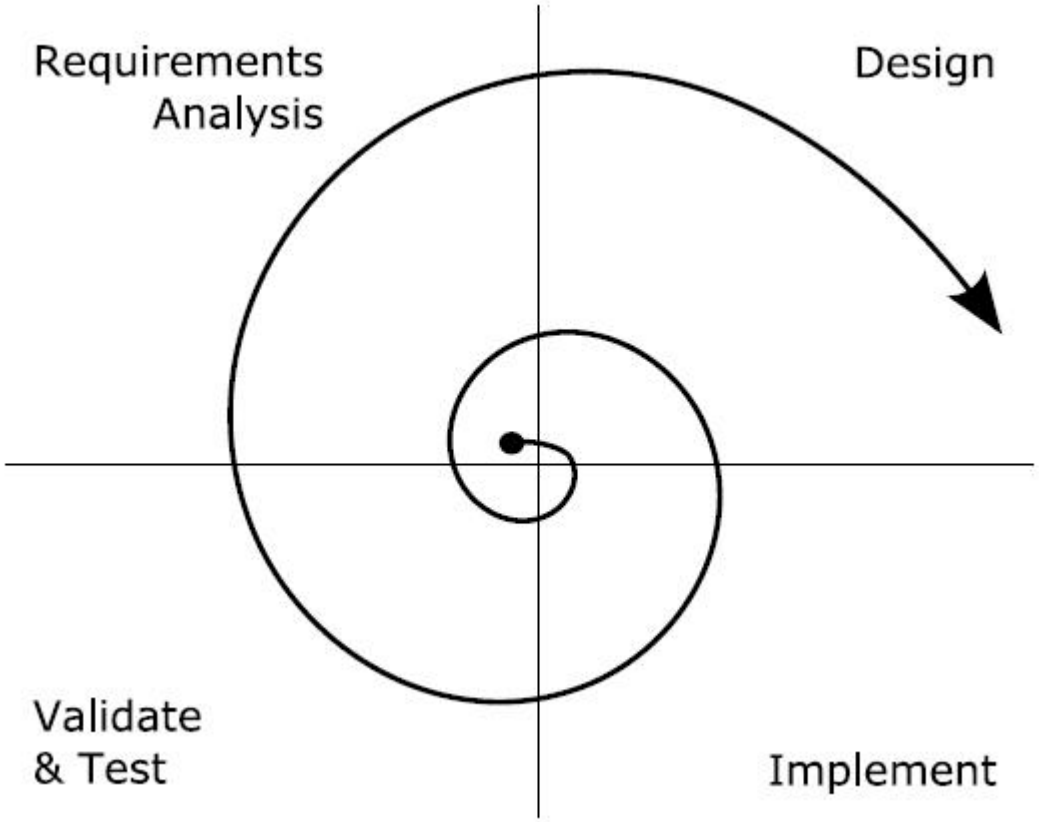
Iterative approach for KEBN

- An iterative and incremental approach for KEBN seems to be a better approach for us

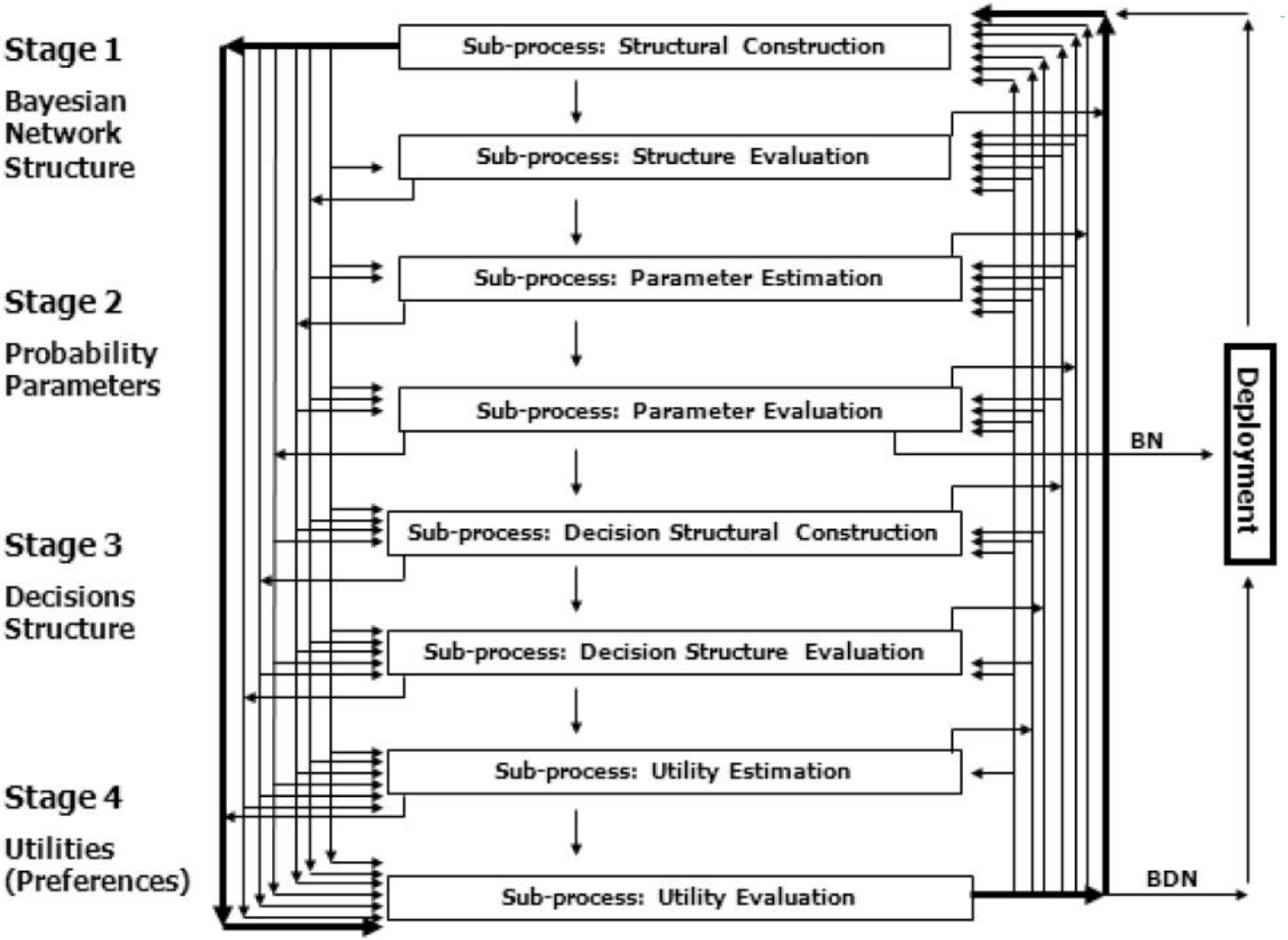
- The software should grow by stages (prototypes) from childhood to adulthood, but at any given stage it is a self-sufficient, if limited, organism.

- Prototypes are functional implementations of software: they accept real input, such as the final system can be expected to deal with, and produce output of the type end-users will expect to find in the final system

A spiral model for KEBN (Korb, Nicholson, 2011)



Iterative lifecyle model for KEBN (Boneh, 2010)



Common mistakes

KEBN aspect	Mistake
The Process	Parameterizing before evaluating structure Trying to build the full model all at once
The Problem	Not understanding the problem context Complexity without value
Structure - Nodes	Getting the node values wrong Node values aren't exhaustive Node values aren't mutually exclusive Incorrect modeling of mutually exclusive outcomes Trying to model fuzzy categories Confusing state and probability Confusion about what the node represents
Structure - Arcs	Getting the arc directions wrong (a) Modeling reasoning rather than causation (b) Inverting cause and effect (c) Missing variables Too many parents
Parameters	Experts' estimates of probabilities are biased (a) Overconfidence (b) Anchoring (c) Availability Inconsistent "filling in" of large CPTs Incoherent probabilities (not summing to 1) Being dead certain

Stage 1: Bayesian Network Structure

- Common Modeling Mistake: Not understanding the problem context

- It is crucial for the knowledge engineer to gain a clear understanding of the problem context. Ideally, this should be available in some form of project description. The knowledge engineer should ask questions like:

Q: *"What do you want to reason about?"*
Q: *"What don't you know?"*
Q: *"What information do you have?"*
Q: *"What do you know?"*

Stage 1: Bayesian Network Structure

- Complexity without value
  - A very common impulse, when something is known about the problem, is to want to put it in the model.
  - But It may add complexity to the model without adding any value (and in fact often reduces value)
- Instead, the knowledge engineer must focus on the

Q: *"Which of the known variables are* most *relevant to the problem?"*

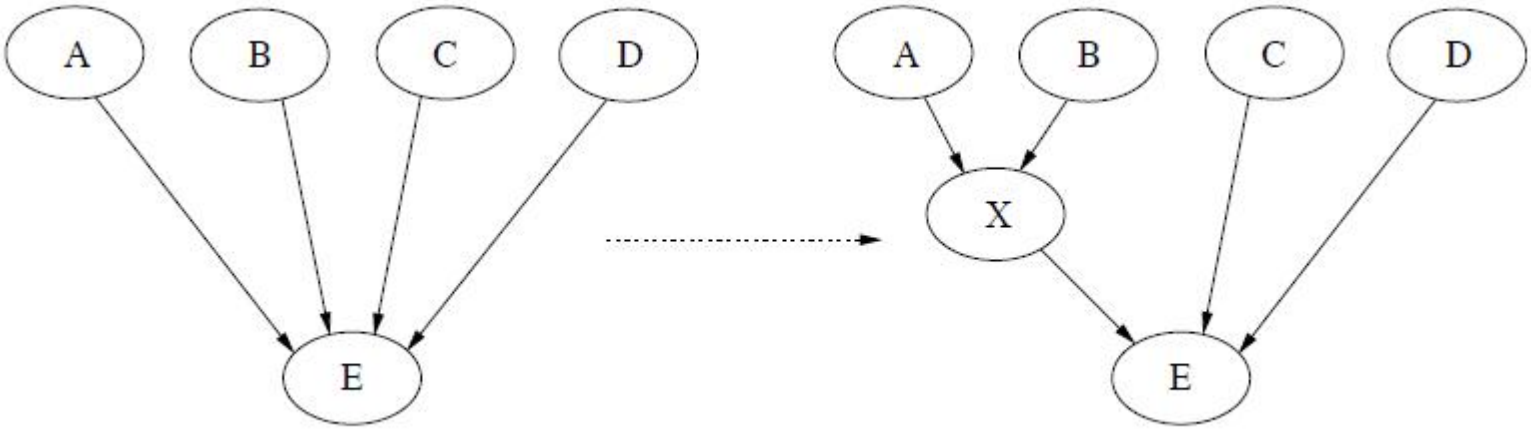
Another commons mistakes in structures

- Getting the node values wrong
- Node values aren't exhaustive
- Node values aren't mutually exclusive
- Incorrect modeling of mutually exclusive outcomes
  - Creation of separate nodes for different states of the same variable. For example, create both a FineWeather variable and a WetWeather variable (both Boolean). They are mutually exclusive!
- Trying to model fuzzy categories
- Confusing state and probability
- Confusion about what the node represents

Stage 1: Bayesian Network Structure

- Other common mistake: Getting the arc directions wrong
  - (a) Modeling reasoning rather than causation
  - (b) Inverting cause and effect
  - (c) Missing variables
- Too many parentes
  - It is usually worse to have many parents than more parents. Modeling new nodes may help…

Reducing parents by intermediate nodes



Discretization

- While it is possible to build BNs with continuous variables without discretization, the simplest approach is to discretize them

- Indeed, many of the current BN software tools available require this. They provides a choice between its doing the discretization for you crudely, (into even-sized chunks) or allowing the knowledge engineer more control over the process

Stage 2: Probability Parameters

- Experts' estimates of probabilities are biased, including
  - Overconfidence: the tendency to attribute higher than justifiable probabilities to events that have a probability sufficiently greater than 0.5.
  - Anchoring: the tendency for subsequent estimates to be biased by an initial estimate (Kahneman and Tversky, 1973)
  - Availability: Assessing an event as more probable than is justifiable, because it is easily remembered or more salient (Kahneman and Tversky, 1973)

Stage 2: Probability Parameters

- Inconsistent "filling in" of large CPTs
  - For example, the expert uses 0.99 for "almost certain" in one part of the CPT, and 0.999 in another. Or it may be inconsistency across the CPT; for example, using different distributions for combinations of parents that in fact are very similar
- Incoherent probabilities (not summing to 1)
- Being dead certain

Stage 3: Decision Structure

- First, we must model what decisions can be made, through the addition of one or more decision nodes.
- If the decision task is to choose only a single decision at any one time from a set of possible actions, only one decision node is required

- Combinations of actions can be modeled within the one node, for example, by explicitly adding a sequence of actions (ex. “surgery-medication”)
  - This modeling solution avoids the complexity of multiple decision nodes, but has the disadvantage that the overlap between different actions

- Alternatively, you can use precedence links or Dynamic BN as you have seen, but you will have to deal with additional complexity!

Stage 4: Utilities (Preferences)

- The next KE task for decision making is to model the utility of outcomes.
- The first stage is to decide what the unit of measure (“utile”) will mean. Remember that money is not equal to utility (but it is related!)
- Remember the process to evaluate utilities of situation through lotteries as we have seen!

Utility Assessment

Utilities map states to real numbers. Which numbers?

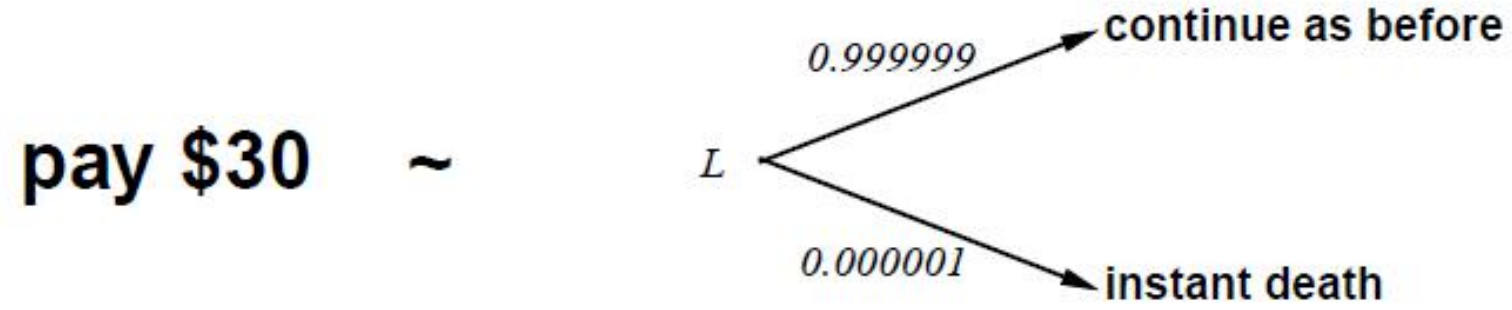
Standard approach to assessment of human utilities:

compare a given state A to a standard lottery L_p that has

"best possible prize" $u_\top$ with probability p

"worst possible catastrophe" $u_\bot$ with probability $(1-p)$

adjust lottery probability p until $A \sim L_p$



Definindo Funções de Utilidades através de loterias

- Dado o intervalo [0,1] entre a "pior catastrofe possível" e "o melhor prêmio possível", ao encontrar uma loteria [p,1;1-p,0] que seja indiferente a uma situação S o número p é a utilidade de S

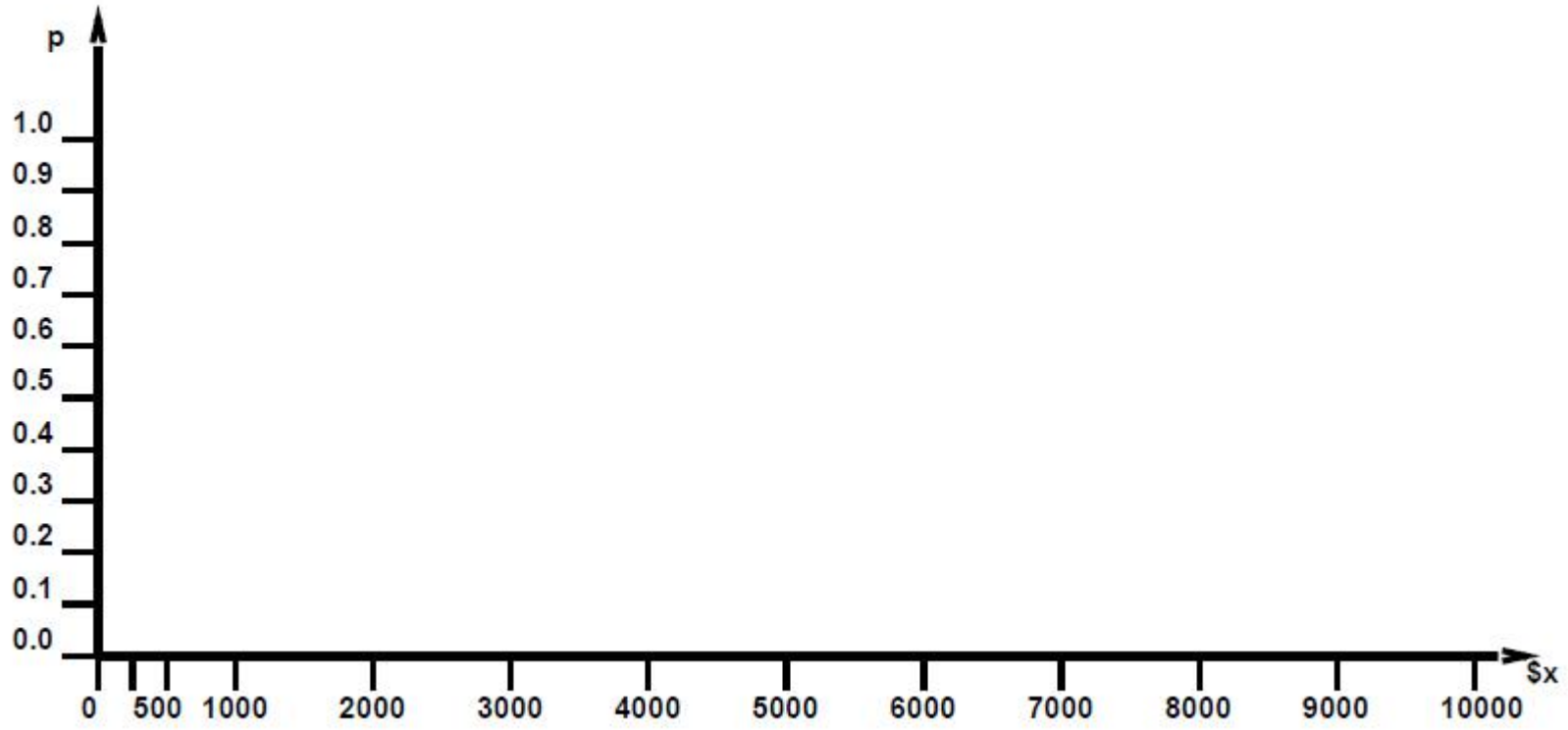
- Em ambientes, com prêmios determinísticos pode-se apenas estabelecer a ordem de preferências, nesse caso usa-se o termo utilidades ordinais

- Funções de utilidades ordinais podem ser chamadas de funções de valor e são invariantes para qualquer transformação monotônica

Utility of the Money

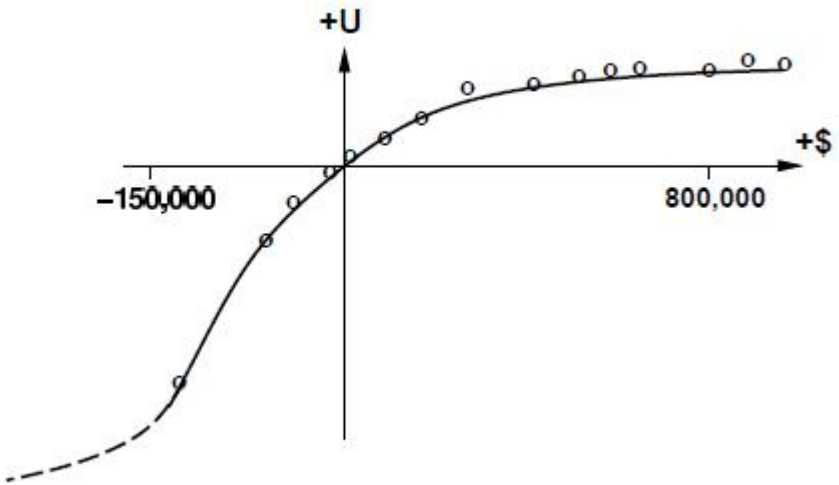
- You choose between take part in the lottery [p,M;1-p,0] or receive the value X for not participating in the lottery

For each x, adjust p until half the class votes for lottery (M=10,000)



Dinheiro vs Utilidade

- Dinheiro não tem uma relação linear (ou simples) com utilidade!
- Ao estimar a utilidade em vários experimentos, observa-se que dada uma loteria L com valor esperado EMV(L) tem-se U(L) < U (EMV(L)), isto é as pessoas são aversas a risco
- Um gráfico típico de dinheiro ($) vs Utilidade (U):



Iterative lifecyle model for KEBN (Boneh, 2010)