Imprecise Probabilities in AI

Credal nets and Probabilistic Circuits

Cassio de Campos (+ F.G.Cozman + A.Antonucci)

c.decampos@tue.nl

TU Eindhoven, The Netherlands

SIPTA Summer School, Bristol, 18 August 2022

Plan

Strong Credal networks

- Generalization of BNs to imprecise probabilities
- Credal sets instead of prob mass functions

  $\{P(X_i|\mathrm{pa}(X_i))\} \Rightarrow \{K(X_i|\mathrm{pa}(X_i))\}$
- Strong (instead of stochastic) independence in the semantics of the Markov condition.
- Convex set of joint mass functions

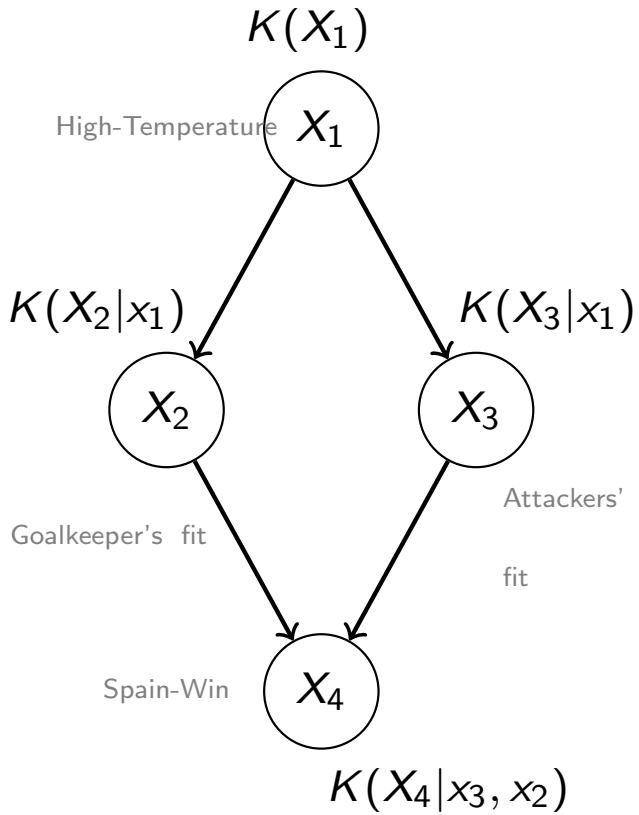
  $K(X_1,\ldots,X_n) = \mathrm{CH}\Big\{P(X_1,\ldots,X_n)\Big\}$

  $P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i|\mathrm{pa}(X_i))$

  $\forall P(X_i|\mathrm{pa}(X_i)) \in K(X_i|\mathrm{pa}(X_i))$

  $\forall i = 1,\ldots,n \quad \forall \mathrm{pa}(X_i)$
- Every conditional mass function takes values in its credal set independently of the others

  CN $\equiv$ (exponential) number of BNs



Local Binary Credal networks

For simplicity, in this presentation we consider locally and separately specified binary credal networks

- Binary means each variable takes only 2 values (typically named true and false).
- Locally means that credal sets involve only the variable of a node and its parents.
- Separately means that credal sets for different parent configurations do not have constraints among them, that is, there is a credal set $K(X_i|pa(X_i))$ for each parent configuration $pa(X_i)$ of the parents of X_i whose constraints are defined over the space of X_i alone.

Local Binary Credal networks

With locally and separately specified binary credal networks:

- Unconditional $K(X_i)$ can be define by two values: $\underline{P}(X_i = true)$ and $\overline{P}(X_i = true)$.
- In general, for each configuration $pa(X_i)$, the set $K(X_i|pa(X_i))$ can be defined by two values $\underline{P}(X_i = true|pa(X_i))$ and $\overline{P}(X_i = true|pa(X_i))$.
- For easy of representation in a data structure, our examples in the next slides use two Bayesian nets with the same graph such that one contains the lower and the other contains the upper value (for each local and separately specified credal set) – note that this is just an implementation trick.

Before we continue with Local Binary Credal networks

- Many constraints can be incorporated into credal networks – this is specially efficient if they are local:
  - Comparative assessments such as $P(X = x'|Y = y) > 2 \cdot P(X = x''|Y = y)$
  - Qualitative assessments such as influences (e.g. Y influences X positively means $P(X = \mathit{true}|Y = \mathit{true}) > P(X = \mathit{true}|Y = \mathit{false})$)

Before we continue with Local Binary Credal networks

- Many constraints can be incorporated into credal networks – this is specially efficient if they are local:
  - Comparative assessments such as $P(X = x'|Y = y) > 2 \cdot P(X = x''|Y = y)$
  - Qualitative assessments such as influences (e.g. Y influences X positively means $P(X = \mathit{true}|Y = \mathit{true}) > P(X = \mathit{true}|Y = \mathit{false})$)
- Non-binary networks can be transformed into binary networks
- Local non-separately specified credal networks can be translated into separated ones

Before we continue with Local Binary Credal networks

- Many constraints can be incorporated into credal networks – this is specially efficient if they are local:
  - Comparative assessments such as $P(X = x'|Y = y) > 2 \cdot P(X = x''|Y = y)$
  - Qualitative assessments such as influences (e.g. Y influences X positively means $P(X = \mathit{true}|Y = \mathit{true}) > P(X = \mathit{true}|Y = \mathit{false})$)
- Non-binary networks can be transformed into binary networks
- Local non-separately specified credal networks can be translated into separated ones
- Non-local constraints/relations require the addition of new auxiliary nodes and may significant increase computational complexity of inferences

(If you want more information, ask Alessandro during the breaks :-))

Credal network - simple example - bn1.txt

```
library(bnlearn)
source('my.bn.inference.r')
net = model2network("[x1][x2|x1][x3|x1][x4|x2:x3]")

cpt1.x1 = matrix(c(0.7, 0.3), ncol = 2,
    dimnames = list(NULL, c('true', 'false')))
cpt1.x2 = c(0.1, 0.9, 0.3, 0.7)
dim(cpt1.x2) = c(2, 2)
dimnames(cpt1.x2) = list("x2" = c("true", "false"),
            "x1" =  c("true", "false"))
cpt1.x3 = c(0.5, 0.5, 0.2, 0.8)
dim(cpt1.x3) = c(2, 2)
dimnames(cpt1.x3) = list("x3" = c("true", "false"),
            "x1" =  c("true", "false"))
cpt1.x4 = c(0.9, 0.1, 0.5, 0.5, 0.4, 0.6, 0.1, 0.9)
dim(cpt1.x4) = c(2, 2, 2)
dimnames(cpt1.x4) = list("x4" = c("true", "false"),
            "x2" =  c("true", "false"),
            "x3" = c("true", "false"))
```

Credal network - simple example - cn1.txt

```
source('my.cn.inference.r')
cpt2.x1 = matrix(c(1, 0), ncol = 2,
    dimnames = list(NULL, c('true', 'false')))

# In this part of the talk, we use a very simplistic
# representation of binary credal networks:
#
# Two BNs, each one gives one of the vertices of
# each local credal set

net.1 = custom.fit(net, dist = list(x1=cpt1.x1,
                x2=cpt1.x2, x3=cpt1.x3, x4=cpt1.x4))
net.2 = custom.fit(net, dist = list(x1=cpt2.x1,
                x2=cpt1.x2, x3=cpt1.x3, x4=cpt1.x4))

# So net.2 is precise apart from variable x1, which
# has 0.7 <= P(x1) <= 1
```

Credal network - simple updating example - cn2.txt

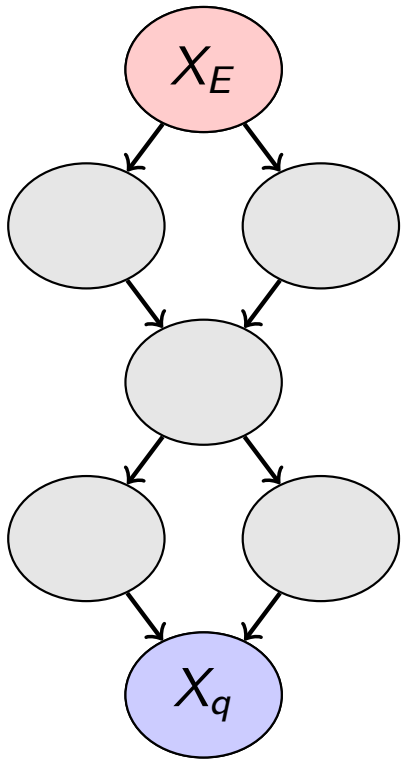
```
query=rep(NA,length(net.1))
names(query) <- names(net.1)
query[2]='false'
res <- my.cn.inference(net.1,net.2,query)
cat('Query p(',res$query,'|',res$evidence,
    ') -- interval result: [',res$min.p,
    ',',res$max.p,']\n')

query[1]='true'
res <- my.cn.inference(net.1,net.2,query)
cat('Query p(',res$query,'|',res$evidence,
    ') -- interval result: [',res$min.p,
    ',',res$max.p,']\n')
```

Updating credal networks

- Conditional probs for a variable of interest X_q given observations $X_E = x_E$
- Updating Bayesian nets is NP-hard
  (fast algorithms for bounded tree-width)

$$P(x_q|x_E) = \frac{P(x_q, x_E)}{P(x_E)} = \frac{\sum_{\mathbf{x}\setminus\{x_q,x_E\}} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{\mathbf{x}\setminus\{x_E\}} \prod_{i=1}^{n} P(x_i|\pi_i)}$$

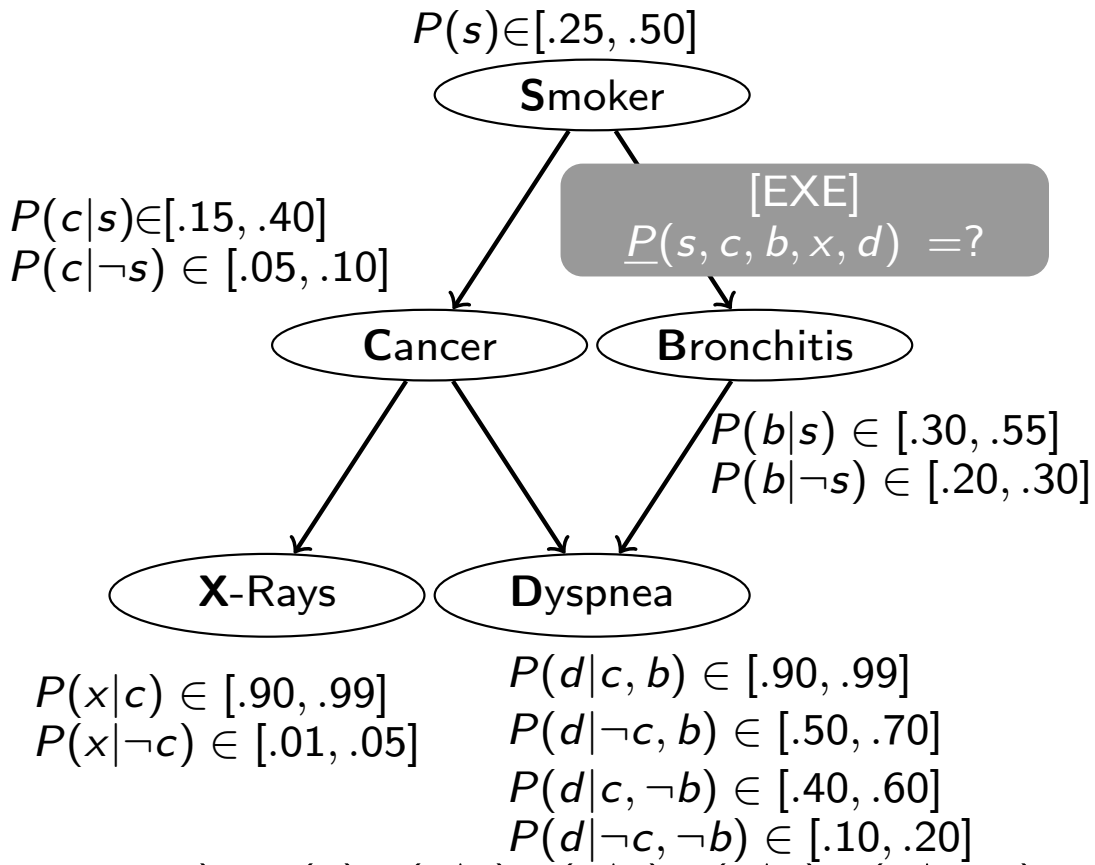


- Updating credal nets is $\mathrm{NP}^{\mathrm{PP}}$-hard,
  NP-hard on polytrees (unless binary and single)
  Easy in trees under epistemic irrelevance

$$\underline{P}(x_q|x_E) = \min_{\substack{P(X_i|\pi_i)\in K(X_i|\pi_i)\\ i=1,\ldots,n}} \frac{\sum_{\mathbf{x}\setminus\{x_Q,x_E\}} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{\mathbf{x}\setminus\{x_Q\}} \prod_{i=1}^{n} P(x_i|\pi_i)}$$

A simple example of diagnosis by CNs

- Five Boolean vars
- Conditional independence relations given by a DAG
- Elicitation of the local (conditional) CSs
- This is a CN specification
- The strong extension $K(S, C, B, X, D) =$



$$\mathrm{CH}\left\{P(S, C, B, X, D) \;\middle|\; \begin{array}{l} P(s, c, b, x, d)=P(s)P(c|s)P(b|s)P(x|c)P(d|c, b) \\ P(S) \in K(S) \\ P(C|s) \in K(C|s), P(C|\neg s) \in K(C|\neg s) \\ \ldots \end{array}\right\}$$

Asia network example - simpleasia1.txt

```
library(bnlearn)

net = model2network("[smoke][lung|smoke][bronc|smoke][xrays|lung][dysp|lung:bronc]")

cpt1.smoke = matrix(c(0.25, 0.75), ncol = 2,
   dimnames = list(NULL, c('true', 'false')))
cpt2.smoke = matrix(c(0.5, 0.5), ncol = 2,
   dimnames = list(NULL, c('true', 'false')))

cpt1.lung = c(0.15, 0.85, 0.05, 0.95)
dim(cpt1.lung) = c(2, 2)
dimnames(cpt1.lung) = list("lung" = c("true", "false"),
"smoke" =  c("true", "false"))
cpt2.lung = c(0.4, 0.6, 0.1, 0.9)
dim(cpt2.lung) = c(2, 2)
dimnames(cpt2.lung) = list("lung" = c("true", "false"),
"smoke" =  c("true", "false"))
cpt1.bronc = c(0.3, 0.7, 0.2, 0.8)
dim(cpt1.bronc) = c(2, 2)
dimnames(cpt1.bronc) = list("bronc" = c("true", "false"),
 "smoke" =  c("true", "false"))
cpt2.bronc = c(0.55, 0.45, 0.3, 0.7)
dim(cpt2.bronc) = c(2, 2)
dimnames(cpt2.bronc) = list("bronc" = c("true", "false"),
 "smoke" =  c("true", "false"))
cpt1.xrays = c(0.9, 0.1, 0.01, 0.99)
dim(cpt1.xrays) = c(2, 2)
dimnames(cpt1.xrays) = list("xrays" = c("true", "false"),
 "lung" =  c("true", "false"))
cpt2.xrays = c(0.99, 0.01, 0.05, 0.95)
dim(cpt2.xrays) = c(2, 2)
dimnames(cpt2.xrays) = list("xrays" = c("true", "false"),
 "lung" =  c("true", "false"))
```

Asia network example - simpleasia2.txt

```
cpt1.dysp = c(0.9, 0.1, 0.5, 0.5, 0.4, 0.6, 0.1, 0.9)
dim(cpt1.dysp) = c(2, 2, 2)
dimnames(cpt1.dysp) = list("dysp" = c("true", "false"),
"lung" =  c("true", "false"), "bronc" = c("true", "false"))
cpt2.dysp = c(0.99, 0.01, 0.7, 0.3, 0.6, 0.4, 0.2, 0.8)
dim(cpt2.dysp) = c(2, 2, 2)
dimnames(cpt2.dysp) = list("dysp" = c("true", "false"),
"lung" =  c("true", "false"), "bronc" = c("true", "false"))
net.1 = custom.fit(net, dist = list(smoke=cpt1.smoke,
  lung=cpt1.lung, bronc=cpt1.bronc, xrays=cpt1.xrays, dysp=cpt1.dysp))
net.2 = custom.fit(net, dist = list(smoke=cpt2.smoke,
  lung=cpt2.lung, bronc=cpt2.bronc, xrays=cpt2.xrays, dysp=cpt2.dysp))
query=rep('true',length(net.1))
names(query) <- names(net.1)
source('my.cn.inference.r')
res <- my.cn.inference(net.1,net.2,query)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
```

Asia network example - simpleasia3.txt

```
query=rep(NA,length(net.1))
names(query) <- names(net.1)
query['lung'] <- 'true'

evi=rep(NA,length(net.1))
names(evi) <- names(net.1)
evi['dysp'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')

evi['bronc'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')

evi['bronc'] <- 'false'
evi['xrays'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
```

Asia network example - simpleasia4.txt

```
evi=rep(NA,length(net.1))
names(evi) <- names(net.1)

evi['bronc'] <- 'false'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
evi['smoke'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
evi['bronc'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
evi['dysp'] <- 'true'
res <- my.cn.inference(net.1,net.2,query,evi)
cat('Query p(',res$query,'|',res$evidence,
   ') -- interval result: [',res$min.p,',',res$max.p,']\n')
```

Plan

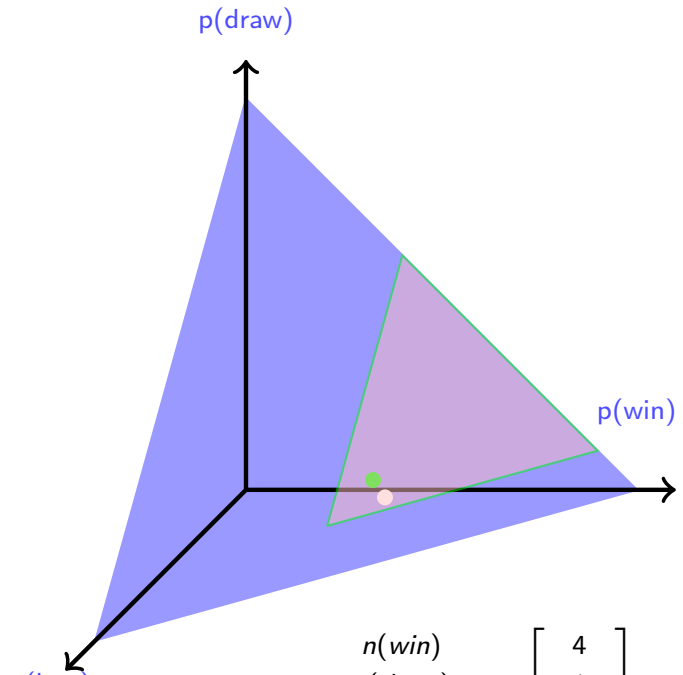
Learning credal sets from (few) data

- Learning from data about X
- Max lik estimate $p(x) = \frac{n(x)}{n}$ (white)
- Bayesian (ESS s) $\frac{n(x)+s\cdot t(x)}{n+s}$ (green)
- Imprecise: set of priors (vacuous t)

$$\frac{n(x)}{n+s} \leq p(x) \leq \frac{n(x)+s}{n+s}$$

imprecise Dirichlet model

- Non-negligible size of intervals only for small n (Bayesian for $n \to \infty$)



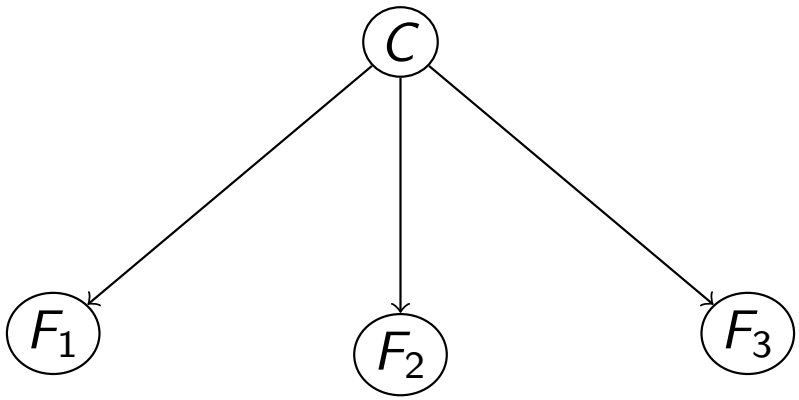
$$\begin{matrix} n(win) \\ n(draw) \\ n(loss) \end{matrix} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}$$

1957: Spain vs. Italy 5 — 1
1973: Italy vs. Spain 3 — 2
1980: Spain vs. Italy 1 — 0
1983: Spain vs. Italy 1 — 0
1983: Italy vs. Spain 2 — 1
1987: Spain vs. Italy 1 — 1
2000: Spain vs. Italy 1 — 2
2001: Italy vs. Spain 1 — 0

Imprecise Dirichlet Model

- The estimates are *imprecise*, being characterized by an upper and a lower probability.
- They do not depend on the sample space.
- The gap between upper and lower probability narrows as more data become available.

Naive Bayes (NBC)



- *Naively* assumes the features to be independent given the class.
- NBC is highly biased, but achieves good accuracy, especially on small data sets, thanks to low variance.
- Learns from data the joint probability of class and features, decomposed as the marginal probability of the classes and the conditional probability of each feature given the class.

Issuing a classification

- The value of the features is specified as $\mathbf{f} = (f_i, \ldots, f_k)$. Then

$$p(c|\mathbf{f}) \propto p(c) \prod_{i=1}^{k} p(f_i|c)$$

where

$$p(c) = \frac{n(c) + s \cdot t(c)}{n + s}$$

$$p(f_i|c) = \frac{n(f_i, c) + s \cdot t(f_i, c)}{n(c) + s \cdot t(c)}.$$

- Prior-dependence: the most probable class varies with $\mathbf{t}$.

Learning credal classifiers

- Induced using a *set* of priors (*credal set*).
- They separate safe instances from prior-dependent ones.
- On prior-dependent instances: they return a set of classes (indeterminate classifications), remaining robust though less informative.

Naive Credal Classifier (NCC)

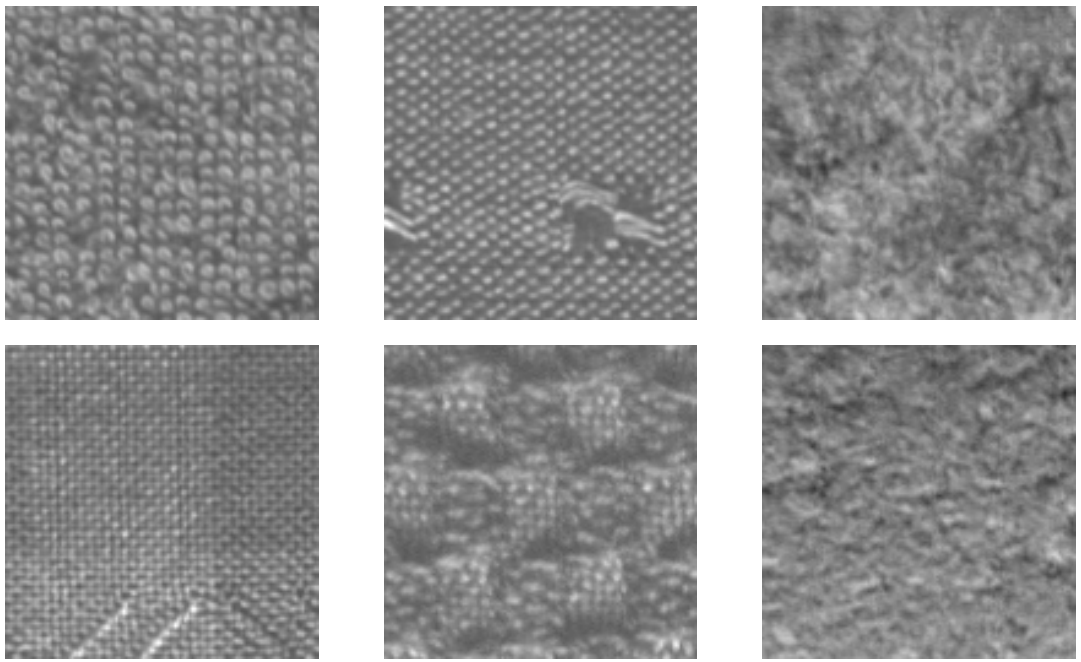
- Uses the IDM to specify a credal set of joint distributions and updates it into a posterior credal set.
- The posterior probability of class c ranges within an *interval*.
- Given feature observation $\mathbf{f}$ and posterior credal set $\mathcal{M}$, class c' dominates c'' if

  $\forall p \in \text{ext}(\mathcal{M}) : \; p(c'|\mathbf{f}) > p(c''|\mathbf{f})$.
- There are other credal criteria (but we do not have time for them).

NCC and prior-dependent instances

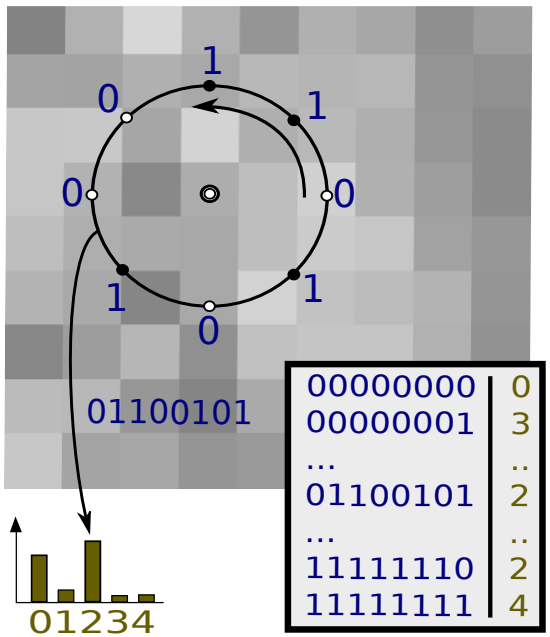
- Credal-dominance is checked by solving an optimization problem, since priors $\mathbf{t}$ are allowed to vary.
- NCC eventually returns the *non-dominated* classes:
  - a singleton on the safe instances
  - a set on the prior-dependent ones.

Texture recognition



- The OUTEX data sets (Ojala, 2002): 4500 images, 24 classes (textiles, carpets, woods ..).

Features: Local Binary Patterns (Ojala, 2002)



- The gray level of each pixel is compared with that of its neighbors, resulting in a binary judgment (more intense/ less intense).
- Such judgments are collected in a string for each pixel.

Results (Corani et al., BMVC 2010)

- Accuracy of NBC: 92% (SVMs: 92.5%).
- NBC is highly accurate on the safe instances, but quite inaccurate on the prior-dependent ones.

	Safe	*Prior-dependent*
Amount%	95%	5%
NBC: accuracy	94%	56%
NCC: set accuracy	94%	85%
NCC: non-dom. classes	1	2.4

Naive Bayes + rejection option vs NCC

- Rejection option: reject an instance (no classification) if the probability of the most probable class is below a threshold p^*.
- But take the case of texture recognition: half of the prior-dependent instances classified by naive Bayes with probability $> 90\%$.
- Rejection rule is not designed to detect prior-dependent instances

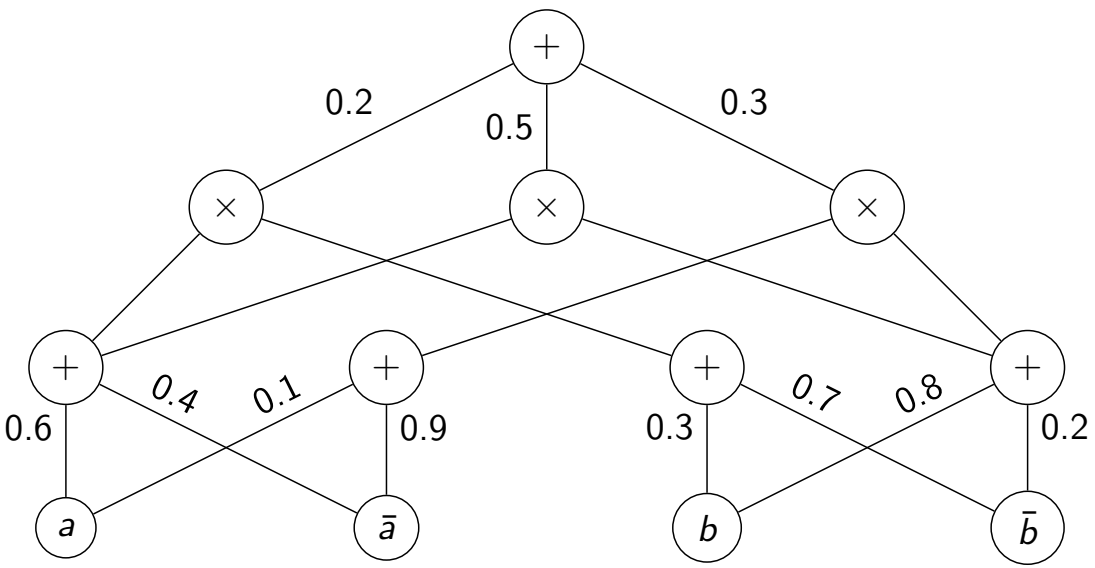
Automatic selecting a classifier

- If we can tell whether we are "certain" about our guess, then we could stop the decision process when we are not very certain.
  - What if your client wants an answer nevertheless? (Give up on imprecise probability? No!)
- We could build the following procedure: For each testing instance,
  - Run a credal classifier, and if "certain", issue a prediction.
  - Run another credal classifier, and if "certain", issue a prediction.
  - Run another credal classifier, and if "certain", issue a prediction.
  - ...

Plan

Deep Models

- Sum-Product Networks: sacrifice "interpretability" for the sake of computational efficiency; represent computations not interactions.
- Complex mixture distributions represented graphically as an arithmetic circuit.



Example (Poon and Domingos 2011)

Learn models from dataset of "faces"; then use it to complete partial face

Sum-Product Network

Distribution $P(X_1, \ldots, X_n)$ built by

- an indicator function over a single variable
  - $I(X = 0)$, $I(Y = 1)$ (also written $\neg x, y$),
- a weighted sum of SPNs with same domain and nonnegative weights
  - $P_3(X, Y) = 0.6 \cdot P_1(X, Y) + 0.4 \cdot P_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $P_3(X, Y, Z, W) = P_1(X, Y) \cdot P_2(Z, W)$.

Sum-Product Network

Distribution $P(X_1, \ldots, X_n)$ built by

- an indicator function over a single variable
  - $I(X = 0)$, $I(Y = 1)$ (also written $\neg x, y$),
- a weighted sum of SPNs with same domain and nonnegative weights
  - $P_3(X, Y) = 0.6 \cdot P_1(X, Y) + 0.4 \cdot P_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $P_3(X, Y, Z, W) = P_1(X, Y) \cdot P_2(Z, W)$.

We can assume that weights are normalized: $\sum_i w_i = 1$.

Sum-Product Network

Distribution $P(X_1, \ldots, X_n)$ built by

- an indicator function over a single variable
  - $I(X = 0)$, $I(Y = 1)$ (also written $\neg x, y$),
- a weighted sum of SPNs with same domain and nonnegative weights
  - $P_3(X, Y) = 0.6 \cdot P_1(X, Y) + 0.4 \cdot P_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $P_3(X, Y, Z, W) = P_1(X, Y) \cdot P_2(Z, W)$.

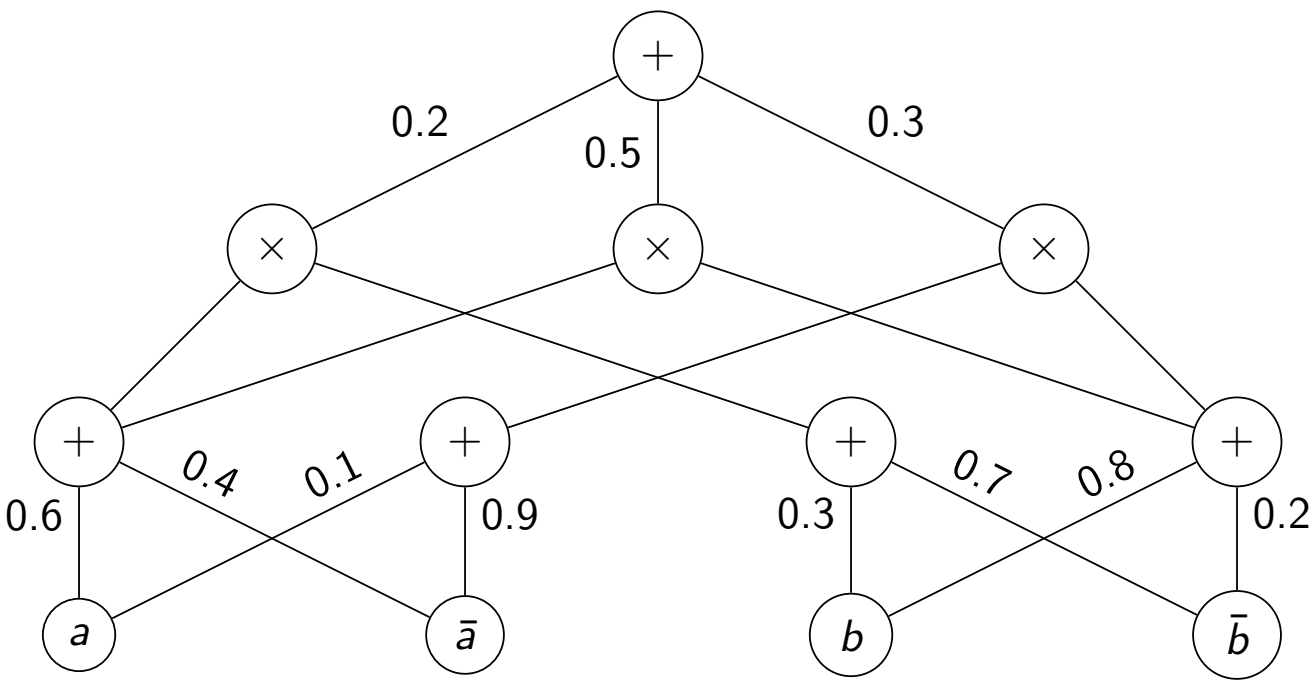
We can assume that weights are normalized: $\sum_i w_i = 1$.

Weighted sums have implicit latent variable:

$0.6 \cdot P_1(X, Y) + 0.4 \cdot P_2(X, Y) = \sum_Z P(Z) \cdot P(X, Y|Z)$.

Graphical Representation

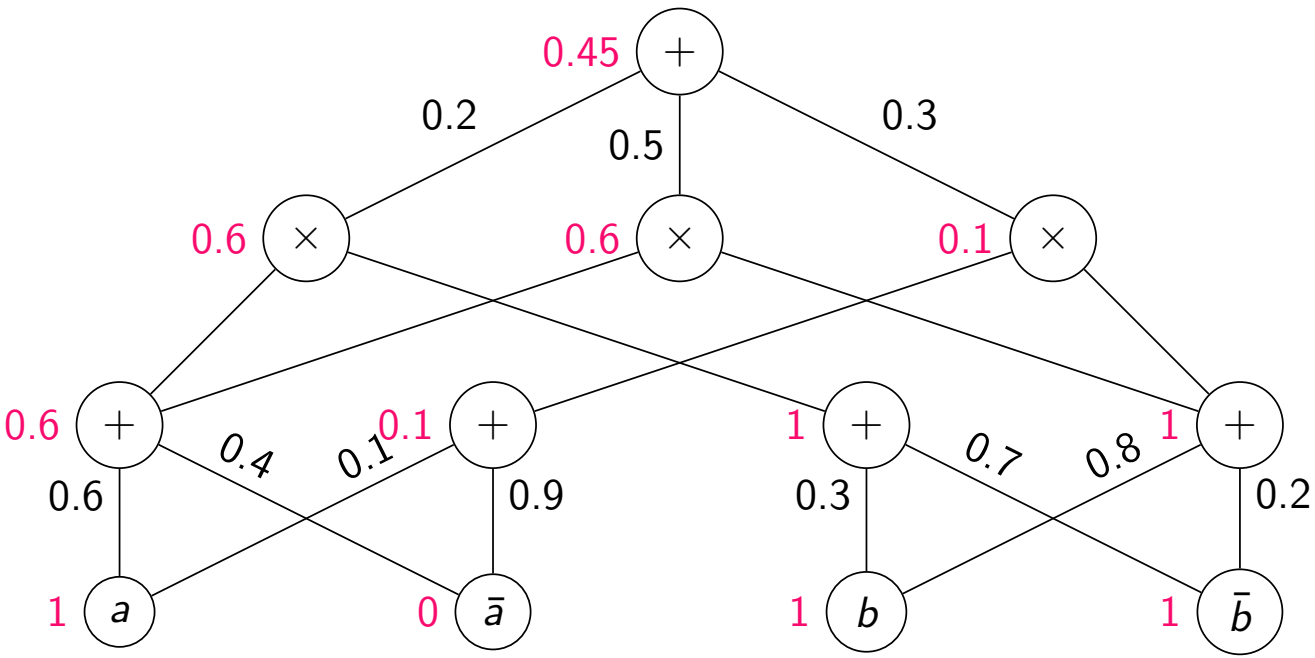
- Rooted directed acyclic graph (directions are implicit below);
- Leaves are indicators;
- Sums and product nodes (edges leaving sum nodes are weighted).



Evaluation (Inference)

- Propagate values bottom-up:

$P(A = 1) =$

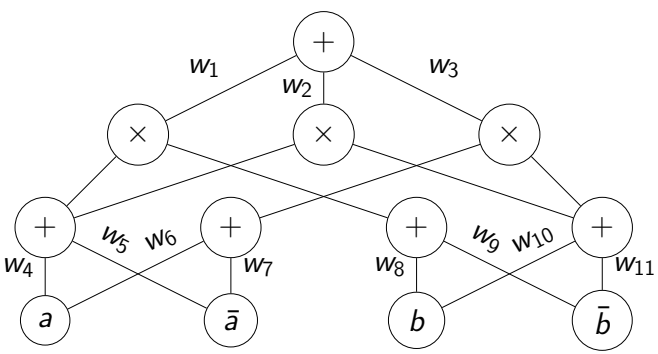


Note: takes linear time in the size of circuit!

Credal Sum-Product Networks

- Robustify SPNs by allowing weights to vary inside sets (for instance, towards sensitivity analisys on SPN's inference).
- New class of tractable imprecise graphical models.

$$\left\{ \begin{array}{c|cc} & (w_1, w_2, w_3) \in \mathrm{CH}(& [0.28, 0.45, 0.27], \\ & [0.18, 0.55, .27], & [0.18, 0.45, 0.37]), \\ & 0.54 \leq w_4 \leq 0.64, & 0.36 \leq w_5 \leq 0.46, \\ & 0.09 \leq w_6 \leq 0.19, & 0.81 \leq w_7 \leq 0.91, \\ & 0.27 \leq w_8 \leq 0.37, & 0.63 \leq w_9 \leq 0.73, \\ & 0.72 \leq w_{10} \leq 0.82, & 0.18 \leq w_{11} \leq 0.28, \\ & w_4 + w_5 = 1, & w_6 + w_7 = 1, \\ & w_8 + w_9 = 1, & w_{10} + w_{11} = 1 \end{array} \right.$$



Credal classification with SPNs

Given configurations c', c'' of variables C and evidence e decide:

$$\min_w \left(P_w(c', e) - P_w(c'', e)\right) > 0.$$

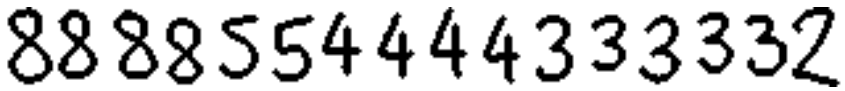
Credal classification with SPNs

Given configurations c', c'' of variables C and evidence e decide:

$$\min_w \left(P_w(c', e) - P_w(c'', e)\right) > 0.$$

Credal classification with a single class variable can be done in polynomial time when each internal node has at most one parent. Also holds if SPN is "class-selective". Many structure learning algorithms generate SPNs of the above form!

Application: Computing robustness index



- Handwritten digit recognition (70 handwritten 20×30 images per digit).
- We learn and check accuracy of an SPN using 50% - 50% train-test split (ramdomly multiple times).
- Robustness index: maximum ϵ s.t. locally ϵ-contaminating weights of SPN does not change classification (that is, single maximal=single e-admissible class).
- Compared against threshold-based robustness (when best - second best probability is below a threshold).

Robustness Measure	CSPN		Best - second best	
	Correct	Wrong	Correct	Wrong
median	0.0363	0.0029	0.0909	0.0880
maximum	0.1524	0.0199	0.3333	0.3333
mean	0.0369	0.0043	0.0976	0.1042

Table: Robustness values. Overall classification accuracy of 99.31%.

SPNs

- Sum-Product Networks offer a recently developed class of probabilistic graphical models with linear time inference.
  - Very promising results in "deep learning": image completion, image classification from pixels, representation learning, etc.
  - Simple (yet powerful) learning methods exist: think about product nodes as independence tests (build a graph and relations and then take connected components), and sum nodes as clustering of data points (as they represent a mixture of distributions).
- Credal Sum-Product Networks extend SPNs to imprecise setting:
  - Weights are associated with sets.
  - Inferences still take polynomial time.
  - Arguably easy to code!

Generative Forests

Adversarial Attacks

- Note that the credal sets and sensitivity analyses discussed so far represent a change of the model.
- Currently it is common to discuss on adversarial attacks, when one perturbs the data point towards fooling the model to make a mistake.
- This can be achieved with credal models by an appropriate trick on how to encode observations in the model
  - One can run a constrained MAP for discrete variables
  - One can take an interval around a continuous observation and propagate the corresponding upper and lower probabilities/densities

Adversarial Attacks

- Note that the credal sets and sensitivity analyses discussed so far represent a change of the model.
- Currently it is common to discuss on adversarial attacks, when one perturbs the data point towards fooling the model to make a mistake.
- This can be achieved with credal models by an appropriate trick on how to encode observations in the model
  - One can run a constrained MAP for discrete variables
  - One can take an interval around a continuous observation and propagate the corresponding upper and lower probabilities/densities
- We will not discuss further today, but the connection is very strong
  - It actually surprises me that we do not see this connection more explicitly mentioned in the literature

Wrap up

- Imprecise probability approaches can be used to stop decision making when in doubt, or to "more confidently" return multiple predictions.
- Robustness can be used to improve classification accuracy.
- Imprecise probability approaches can deal with missing data in a conservative manner.
- Imprecise models can be as simple as the Naive Bayes and as complicated as general Bayesian networks. Deep learning is also on the table.
- There are multiple cases where the imprecise analogue of a (precise) model has no (or minimal) loss in terms of computational complexity!

Some references

- Antonucci, A., de Campos, C.P., Zaffalon, M. (2014). *Probabilistic graphical models*. In Augustin, T., Coolen, F., de Cooman, G., Troffaes, M. (Eds), Introduction to Imprecise Probabilities, Wiley, pp. 207–229.
- Bernard, J-M. (2005). *An introduction to the imprecise Dirichlet model for multinomial data*. International Journal of Approximate Reasoning 39 (2–3), pp. 123–150.
- De Bock, J., de Campos, C.P., Antonucci, A. (2014). *Global sensitivity analysis for MAP inference in graphical models*. Advances in Neural Information Processing Systems 27 (NIPS), pp. 2690–2698.
- de Campos, C. P., Cozman, F. G. (2005). *The inferential complexity of Bayesian and credal networks*, Int. Joint Conference on Artificial Intelligence (IJCAI), 5 pp. 1313-1318.

Some references

- Conaty, D., de Campos, C., Martinez Del Rincon, J. (2018). *Cascading Sum-Product Networks using Robustness*. Int. Conference on Probabilistic Graphical Models (PGM), PMLR.
- de Cooman, G., Zaffalon, M. (2004). *Updating beliefs with incomplete observations*. Artificial Intelligence 159(1–2), pp. 75–125.
- Corani, G., Giusti, A., Migliore, D., Schmidhuber, J. (2010). *Robust Texture Recognition Using Credal Classifiers*. British Machine Vision Conference (BMVC), pp. 78.1–78.10.
- Cozman, F. G. (2000). *Credal networks*. Artificial Intelligence 120 (2), pp. 199–233.
- Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks* Cambridge Press.
- Correia, A. H. C., Peharz, R., de Campos, C. P. (2020).*Joints in Random Forests*. NeurIPS 2020.

Some references

- Gens, R., Domingos P. (2013). *Learning the Structure of Sum-Product Networks*. International Conference on Machine Learning (ICML), PMLR 28(3):873-880, 2013.
- Maua, D.D., Conaty, D., Cozman, F.G., Poppenhaeger, K., de Campos, C.P. (2018). *Robustifying sum-product networks*. International Journal of Approximate Reasoning, in press (also at ISIPTA'17).
- Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.
- Poon, H., Domingos, P. (2011). *Sum-product networks: A new deep architecture* IEEE International Conference on Computer Vision Workshops.
- Zaffalon, M., Miranda, E. (2009). *Conservative Inference Rule for Uncertain Reasoning under Incompleteness* Journal of Artificial Intelligence Research 34, pp. 757–821.
- Zaffalon, M. (2002) *The naive credal classifier*. Journal of Statistical Planning and Inference, 105(1), pp. 5–21.