(Some) Machine Learning

with (some) Imprecise Probabilities

Cassio P. de Campos

c.decampos@uu.nl

Utrecht University, The Netherlands

SIPTA Summer School, University of Oviedo, July 2018

Plan

Motivating Imprecise Probability

- Treatment of missing data
- Reliable classification
- Sensitivity analysis
- Feature selection – "robust" statistical tests
- It can be fast!

R

- *http://www.r-project.org*
- *http://www.bombonera.org/siptaschool2018/*

- `install.packages("e1071")`
- `install.packages("bnlearn")`
- `install.packages("IDPSurvival")`
- `install.packages("cluster")`
- `install.packages("igraph")`
- `install.packages("ROCR")`

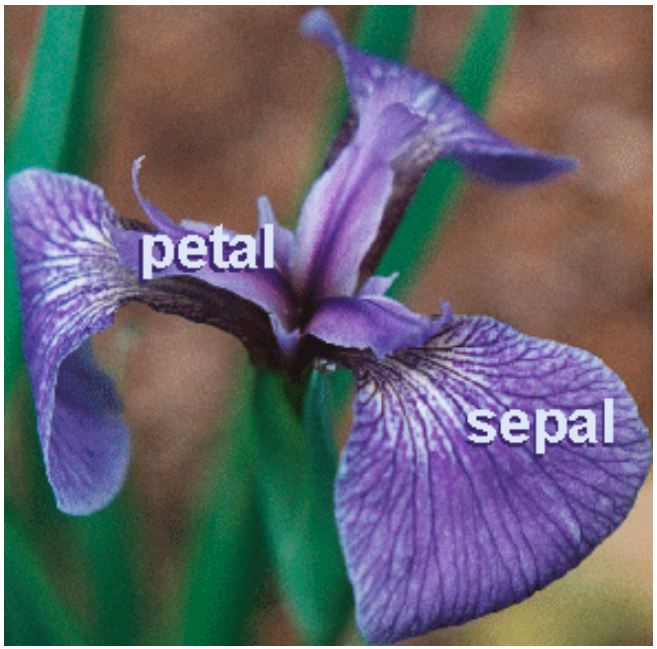
Reliable classification

- Consider the following problem:
  - Objects contain some defining features (say m of them) that (possibly) can be used to identify them.
  - Objects can be categorized into classes. The class of an object might be unknown to us.
- Given a collection of objects of known classes, build a model that can "guess" the class of an object of unknown class.
- Let us assume a log-linear model (C class var, F_i features):

$$P(C|F_1,\ldots,F_m) \propto P(C)\cdot\prod_{i=1}^{m} P(F_m|C)$$

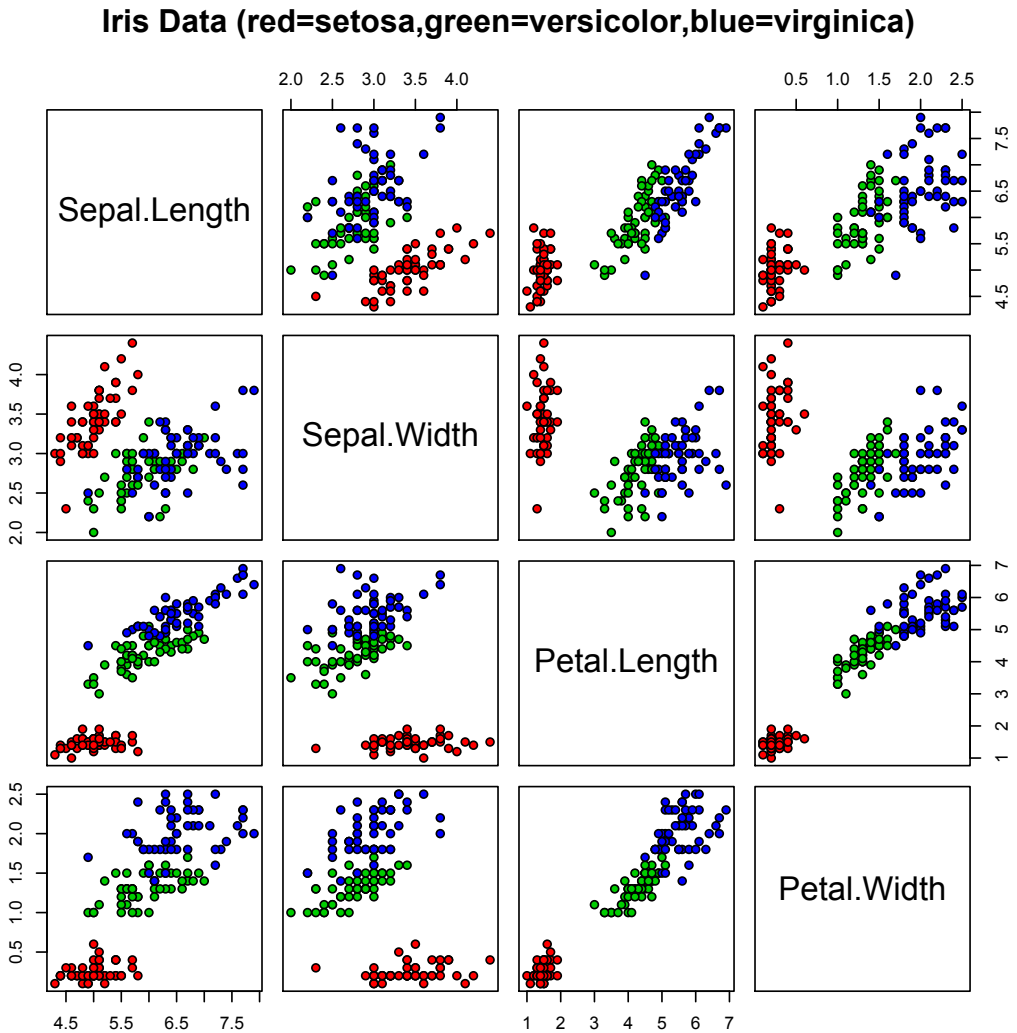
- Given $F_1,\ldots,F_m$, guessing the class can be done by taking $\max_C P(C|F_1,\ldots,F_m)$.
- Values $P(C)$ and $P(F_i|C)$ can be inferred using the collection of objects of known classes.

Reliable classification - Iris example



http://mirlab.org/jang/books/dcpr/image/iris.gif

Reliable classification - Iris example



Iris example - classification1.txt

```
options(width=300)
library(e1071)
data(iris)
#pairs(iris[1:4],
#  main="Iris Data (R=setosa,G=versicolor,B=virginica)",
#  pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])
for(i in 1:ncol(iris)) if(is.numeric(iris[,i]))
  iris[,i] <- as.factor(iris[,i] > median(iris[,i]))
iris[1:10,]
iris.training <- iris[c(1:30,51:80,101:130),]
iris.testing <- iris[-c(1:30,51:80,101:130),]
```

Iris example - classification2.txt

```
model <- naiveBayes(Species ~ ., laplace=1,
  data = iris.training)
prediction.classes <- predict(model, iris.testing)
prediction.probs <- predict(model, iris.testing, type='raw')
probs.max <- apply(prediction.probs, 1, max)

sum(prediction.classes == iris.testing$Species)/
   length(iris.testing$Species)
table(prediction.classes,iris.testing$Species)
summary(prediction.classes)
```

Iris example - classification3.txt

```
cut <- 0.95
prediction.classes.high <- prediction.classes[probs.max > cut]
sum(prediction.classes.high ==
    iris.testing$Species[probs.max > cut])/
    length(prediction.classes.high)
table(prediction.classes.high,
        iris.testing$Species[probs.max > cut])

prediction.classes.low <- prediction.classes[probs.max <= cut]
sum(prediction.classes.low ==
     iris.testing$Species[probs.max <= cut])/
     length(prediction.classes.low)
table(prediction.classes.low,
     iris.testing$Species[probs.max <= cut])
```

Iris example - classification4.txt

```
pred <- prediction(probs.max, prediction.classes == iris.testing$Species)  #<--
##                     REALITY (for us means got correct class!)
##                     1       0
##           1        TP      FP          TP/(TP+FP) = accuracy above thr
##  GUESS    0        FN      TN          FN/(TN+FN) = accuracy below thr

##CAN PROB TELL IF WE WILL GUESS RIGHT OR WRONG?
perf = performance(pred, measure = "acc")
plot(perf)
perf = performance(pred, measure = "ppv") #<------
x=rev(unlist(perf@x.values)[-1])
y=rev(unlist(perf@y.values)[-1])
plot(stepfun(x,c(y,1),right=T))

## ACCURACY ABOVE VS ACCURACY BELOW (weird plot)
perf <- performance(pred, measure = "ppv", x.measure = "pcmiss")
plot(perf, colorize=T, print.cutoffs.at=perf@alpha.values, lwd=3,
       text.adj=c(1.2,1.2))
```

Iris example - classification5.txt

```
data(iris)
for(i in 1:ncol(iris)) if(is.numeric(iris[,i]))
  iris[,i] <- as.factor(iris[,i] > median(iris[,i]))
iris$Species = (iris$Species == 'virginica')

source('classification.r')
myclassifier = classifier.composed(
      list(classifier.naive2(0),
           classifier.naive2(1),
           classifier.credal(5)))
print(kfcv.classifier(iris, 1:4, 5, myclassifier))
```

Reliable classification

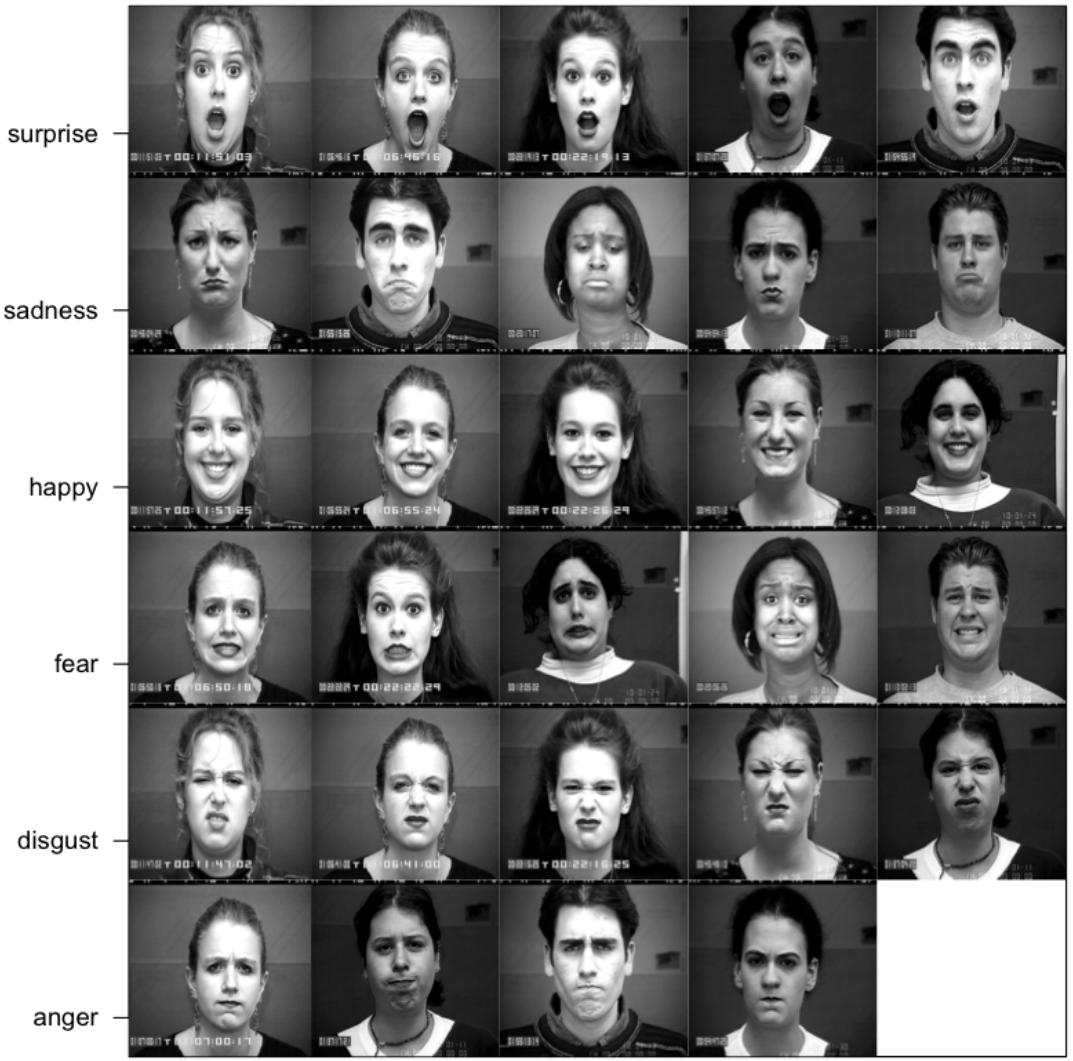
- Can we improve classification accuracy?
- Can we provide a subset of the classes that contains the correct one?
- Can we identify hard- and easy-to-classify instances?

- If probabilities are wrong, a simple cut-off or rejection rule might not be enough.

Sensitivity Analysis

- Suppose that using a probabilistic model, we have reached a conclusion. Is this conclusion sensitive to modifications of the model?
- Usual procedure is to apply local modifications to the model and to check whether the conclusion remains inaltered.

Sensitivity Analysis - an example

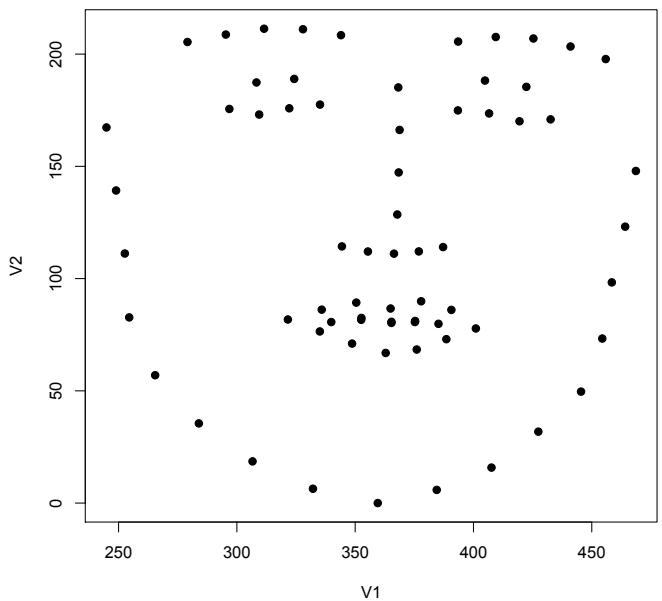


Cohn-Kanade (CK+) database, CVPR 2010.

If you can, try to identify the best expression representing what Mr E thinks of Penelope.

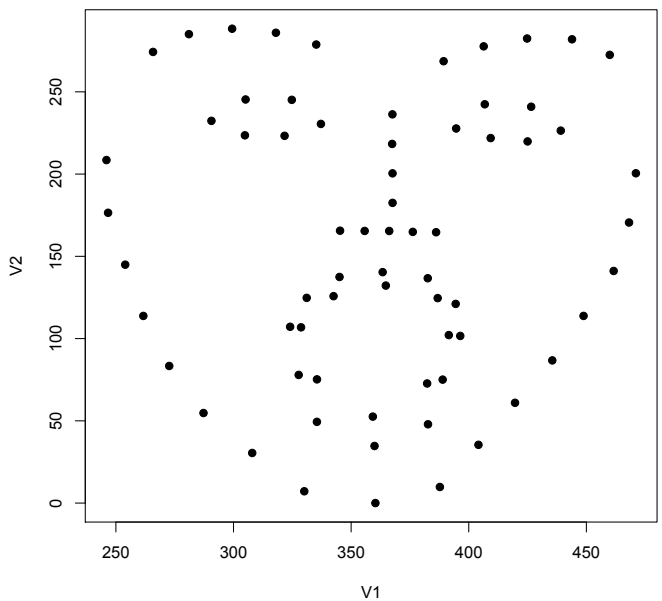
Sensitivity Analysis - an example

Anger



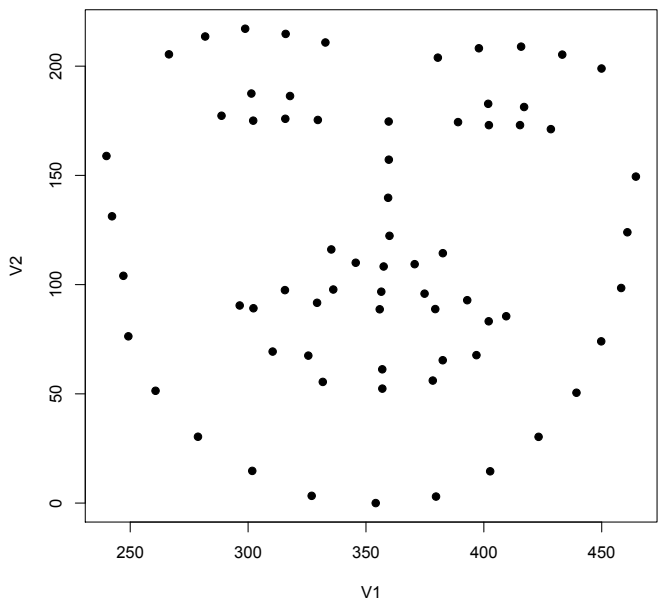
Sensitivity Analysis - an example

Surprise



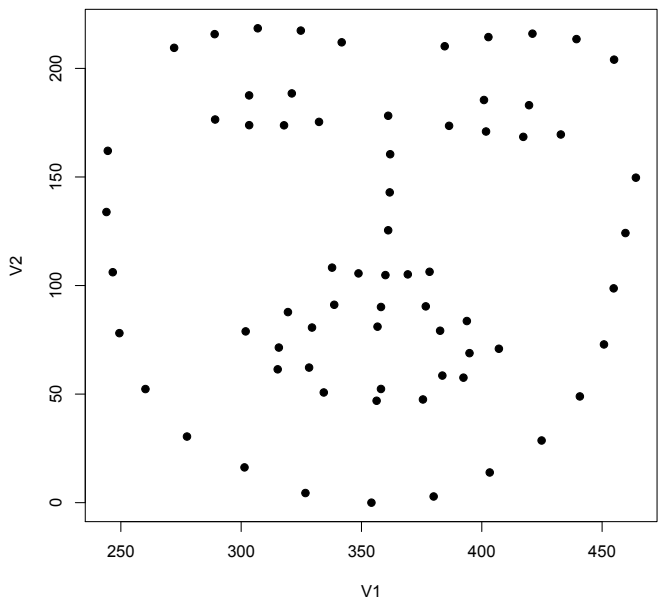
Sensitivity Analysis - an example

Happy



Sensitivity Analysis - an example

Fear



Sensitivity Analysis - an example

- We build a model with 23 Facial action units (facs) from the landmarks.
- We predict all these 23 facs.
- Standard techniques achieve about 90% accuracy.

AU Number	FACS Name	Muscular Basis
0	Neutral face	
1	Inner Brow Raiser	frontalis (pars medialis)
2	Outer Brow Raiser	frontalis (pars lateralis)
4	Brow Lowerer	depressor glabellae, depressor supercilii, corrugator supercilii
5	Upper Lid Raiser	levator palpebrae superioris, superior tarsal muscle
6	Cheek Raiser	orbicularis oculi (pars orbitalis)
7	Lid Tightener	orbicularis oculi (pars palpebralis)
8	Lips Toward Each Other	orbicularis oris
9	Nose Wrinkler	levator labii superioris alaeque nasi
10	Upper Lip Raiser	levator labii superioris, caput infraorbitalis

Source: wikipedia.

Sensitivity Analysis

- The model for expression recognition is quite sophisticated. Are the results reliable?
- If we employ a small modification in one parameter, would results change?
- If we allow all model parameters to vary within a region (near the estimated values), would results change?
- Some facial expressions are arguably easier to spot. Can we automatically identify that fact?

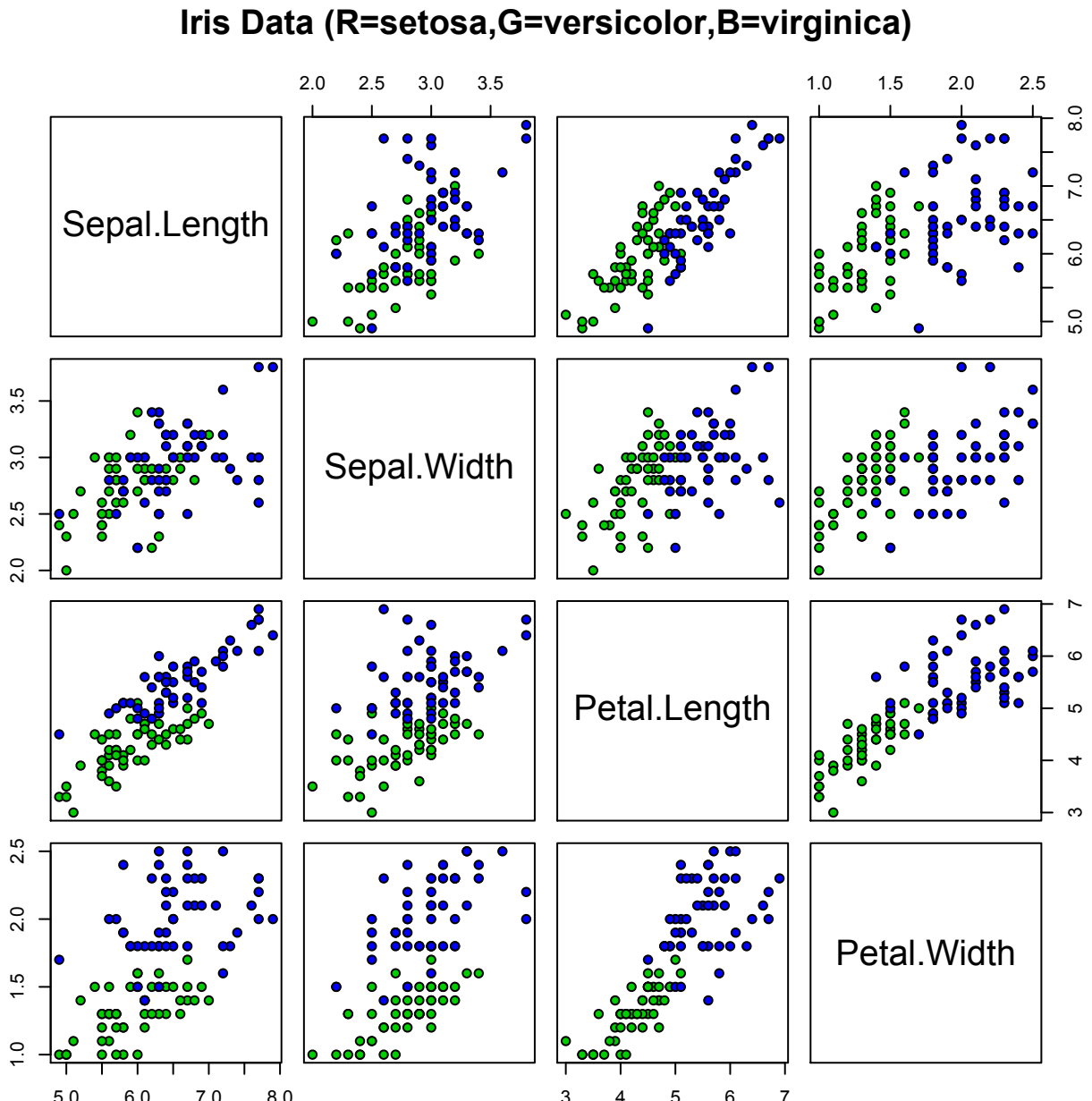
"Robust" feature selection

- We are given a (potentially large) number of covariates and want to identify those which are useful to predict a binary response.
- For example, let us choose only some F_i to include in the model:

$$P(C|F_1,\ldots,F_m) \propto P(C)\cdot\prod_{i=1}^{m} P(F_m|C)$$

- An usual procedure is to employ some statistical tests (sign-rank, rank-sum test, t-test, etc).
- *Mann-Whitney is more robust than the Student's t-test*, Wikipedia [citation needed]. "If it is written on wikipedia, then it is true", anonymous author.

"Robust" feature selection - an example



"Robust" feature selection - an example - feature1.txt

```
data(iris)
iris.versicolor <- iris[iris$Species == 'versicolor',]
iris.virginica <- iris[iris$Species == 'virginica',]
n = nrow(iris.virginica)

set.seed(1)
g1 <- sample.int(n,n/2)
g2 <- sample.int(n,n/2)

wilcox.test(iris.versicolor$Sepal.Width[g1],
   iris.virginica$Sepal.Width[g2],alternative='less')
wilcox.test(iris.versicolor$Sepal.Width[-g1],
   iris.virginica$Sepal.Width[-g2],alternative='less')
```

"Robust" statistical tests

- Can we tell whether the result of a statistical test is robust or not?
- Are usual tests calibrated?
- Can we come up with a measure of "robustness" for the test result?

Plan

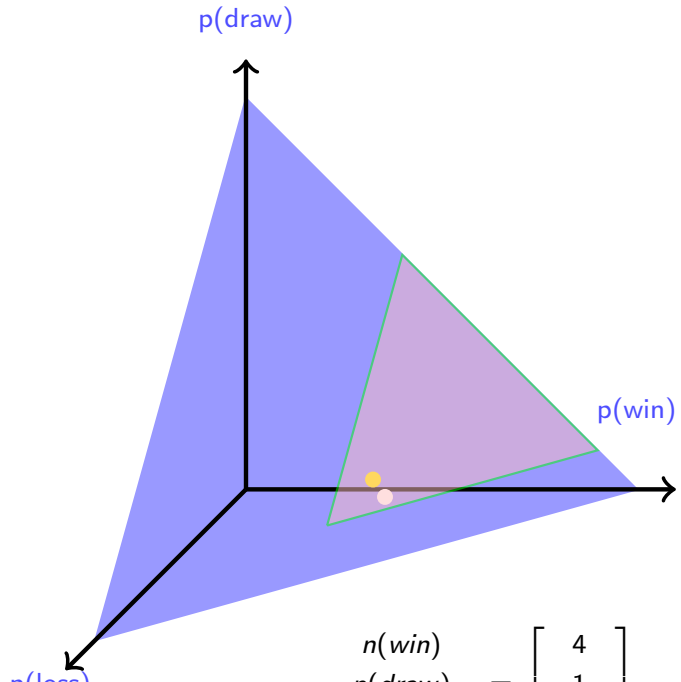
Learning credal sets from (few) data

- Learning from data about X
- Max lik estimate $p(x) = \frac{n(x)}{n}$
- Bayesian (ESS s) $\frac{n(x)+s \cdot t(x)}{n}$
- 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 $n \to \infty$)



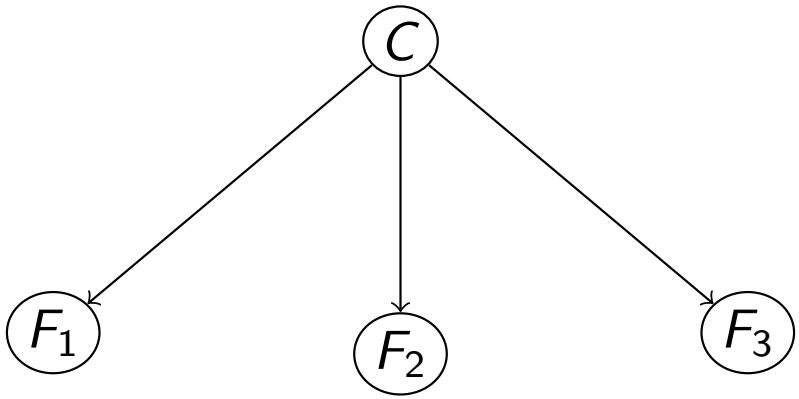
$$\begin{matrix} n(win) \\ n(draw) \\ n(loss) \end{matrix} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}$$

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 $\mathbf{t}$.

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)

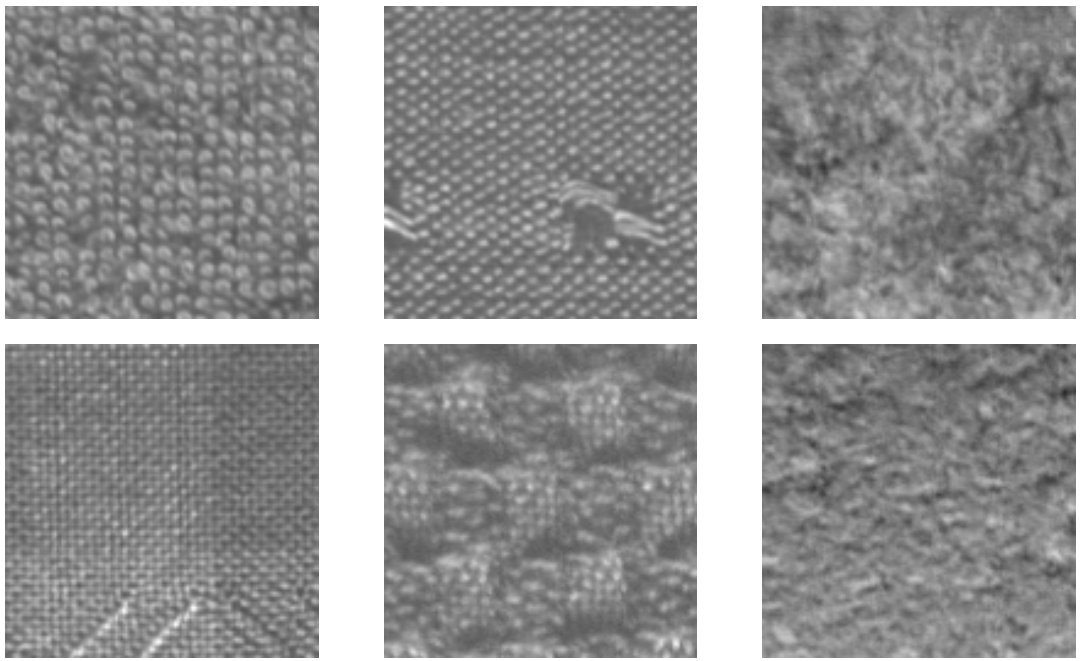
- Uses the IDM to specify a credal set of joint distributions and updates it into a posterior credal set.
- The posterior probability of class c ranges within an *interval*.
- Given feature observation $\mathbf{f}$ and posterior credal set $\mathcal{M}$, class c' credal-dominates c'' if

  $\forall p \in \mathcal{M}: \; p(c'|\mathbf{f}) > p(c''|\mathbf{f})$.
- Which one is this decision making criterion?

NCC and prior-dependent instances

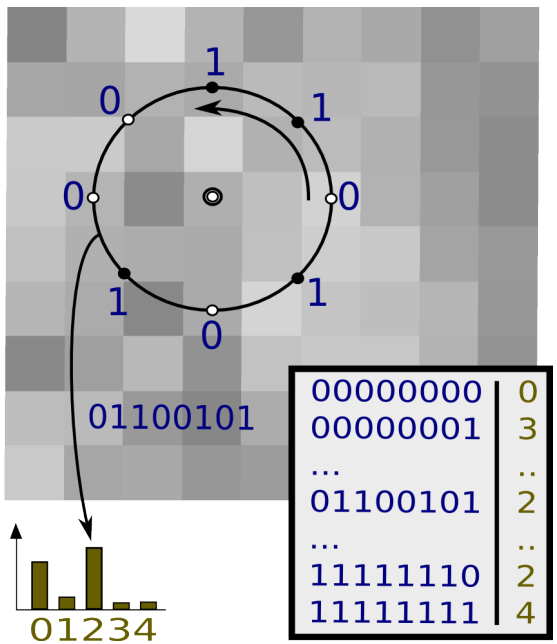
- Credal-dominance is checked by solving an optimization problem, since priors $\mathbf{t}$ are allowed to vary.
- NCC eventually returns the *non-dominated* classes:
  - a singleton on the safe instances
  - a set on the prior-dependent ones.

Texture recognition



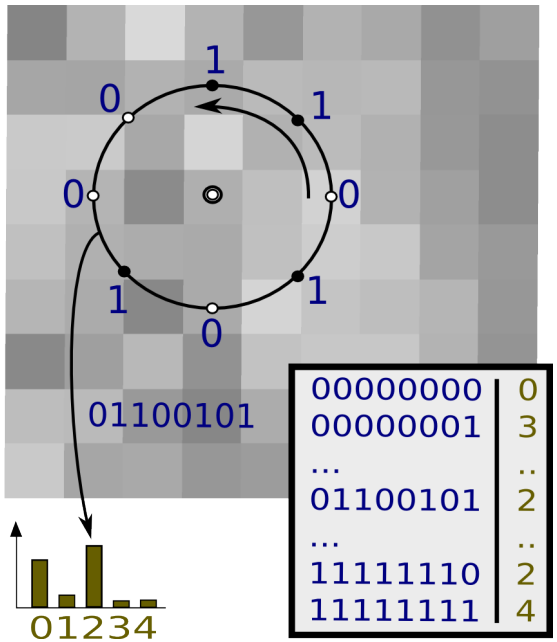
- The OUTEX data sets (Ojala, 2002): 4500 images, 24 classes (textiles, carpets, woods ..).

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.

Local Binary Patterns (2)



- Each string is then assigned to a single category.
- The categories group similar strings: e.g., 00001111 is in the same category of 11110000 for rotational invariance.
- There are 18 categories.
- For each image there are 18 features: the % of pixels assigned to each category.

Results (Corani et al., BMVC 2010)

- Accuracy of NBC: 92% (SVMs: 92.5%).
- NBC is highly accurate on the safe instances, but almost random 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^*. (We have done this in the motivation examples.)
- 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!

NCC and reachable intervals

- Credal set with reachable intervals: *picking an upper for some category and the lower of another* is an element of the credal set.
- For a given test instance $\mathbf{f}$ (let us assume that $\underline{p}(\mathbf{f}) > 0$),

$$\begin{aligned}
&\forall p \in \mathcal{M}: \; p(c'|\mathbf{f}) > p(c''|\mathbf{f}) \iff \\
&\forall p \in \mathcal{M}: \; p(c', \mathbf{f}) > p(c'', \mathbf{f}) \iff \\
&\forall p \in \mathcal{M}: \; p(c') \cdot \prod_i p(f_i|c') > p(c'') \cdot \prod_i p(f_i|c'') \iff \\
&\qquad \underline{p}(c') \cdot \prod_i \underline{p}(f_i|c') > \overline{p}(c'') \cdot \prod_i \overline{p}(f_i|c'')
\end{aligned}$$

- This is the case, for instance, with binary classification.
- Thanks Matthias, we can use `classification.r`

Learning NCC with local IDM

- If we assume that we are using a local IDM to learn the model, then:

$$\underline{p}(c') \cdot \prod_i \underline{p}(f_i|c') = \frac{n(c')}{n+s} \prod_i \frac{n(f_i, c')}{n(c')+s}$$

$$\overline{p}(c'') \cdot \prod_i \overline{p}(f_i|c'') = \frac{n(c'')+s}{n+s} \prod_i \frac{n(f_i, c'')+s}{n(c'')+s}$$

- With a global IDM, things are more complicated (but still efficient, see the NCC2).

Robustness measure

- A (prior-based) robustness measure can be devised based on the largest possible set of priors which still allows for precise decision (this measure is based on the test instance, but no need to know the true label).
- One could create a procedure which finds the maximum *equivalent sample size* for IDM under such constraint (of single prediction).
- Such measure could be used to improve classification accuracy.

Robustness measure

- A (prior-based) robustness measure can be devised based on the largest possible set of priors which still allows for precise decision (this measure is based on the test instance, but no need to know the true label).
- One could create a procedure which finds the maximum *equivalent sample size* for IDM under such constraint (of single prediction).
- Such measure could be used to improve classification accuracy.
- Not so easy to use `classification.r` though, but we are brave!

Robustness measure – 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

Missing Data – an example

- Patient presents symptoms that could be related to lung cancer.
- Physician can run tests for *Bronchitis* and do *X-rays*, as well as check for *Dyspnea*. However, (supposedly) they can only assess whether the patient is a *Smoker* by asking the patient themselves.
- Patient did not answer whether they are a smoker in the questionnaire.
- (Hidden information: patient has a discount in their insurance because they declared *not* to be a smoker to the insurance company.)

Should *smoking* be ignored? Should it be marginalized out?
Should it be treated with (greater) care?

Missing Data – another example

- Suppose we are given the following questionnaire.
- The possible answers are *bad*, *so-so*, *good*. It is also possible to leave it empty.

	Vlad	Barack	Roger	Maria	Penelope

Missing Data – another example

- Suppose we are given the following questionnaire.
- The possible answers are *bad*, *so-so*, *good*. It is also possible to leave it empty.

	Vlad	Barack	Roger	Maria	Penelope
Mr E					
Mr A					
Mr C					

Let's fill it in!

Missing Data – another example

- Earlier today I gave the following questionnaire to three people, whose identity shall be kept anonymous.
- The only answer options are *bad* or *so-so*. It is also possible to leave it empty.

	Vlad	Barack	Roger	Maria	Penelope
Mr E	bad	so-so			bad
Mr A	bad		so-so		so-so
Mr C	bad	bad	so-so	good	

Missing Data – another example

- Earlier today I gave the following questionnaire to three people, whose identity shall be kept anonymous.
- The only answer options are *bad* or *so-so*. It is also possible to leave it empty.

	Vlad	Barack	Roger	Maria	Penelope
Mr E	bad	so-so	GREAT	AMAZING	bad
Mr A	bad	GOOD	so-so	WHO IS?	so-so
Mr C	bad	bad	so-so	good	GREAT

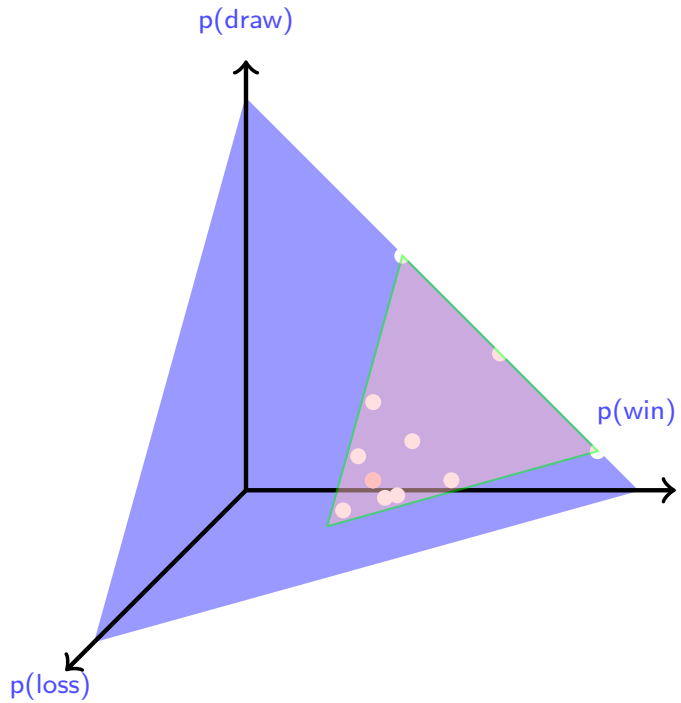
"The only way to obtain a proper estimation is to model missingness."

Missing Data

- Should we consider missing as another category?
- Should we consider all possible completions of the data?
- Should we treat some missing values in a way, some in another?

Learning credal sets from (missing) data

- Coping with missing data?
- Missing at random (MAR): Ignore missing data
- Not always the case!



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
2003:	Spain	vs.	Italy	$*$ — $*$
2011:	Italy	vs.	Spain	$*$ — $*$

Ignorance from missing data

- Usually, classifiers *ignore* missing data, assuming them to be MAR (missing at random).
- MAR: the probability of an observation to be missing does not depend on its value or on the value of other missing data.
- The MAR assumption cannot be tested on the incomplete data.

A non-MAR example: a political poll.

- The right-wing supporters (blue) sometimes refuse to answer; left-wing (red) supporters always answer.

Vote	Answer
red	red
red	red
red	red
blue	blue
blue	-
blue	-

- By ignoring missing data, $p(blue) = 1/4$: underestimated!

Conservative treatment of missing data.

- Consider each possible completion of the data.

Answer	**D1**	**D2**	**D3**	**D4**
red	red	red	red	red
red	red	red	red	red
red	red	red	red	red
blue	blue	blue	blue	blue
-	red	red	blue	blue
-	red	blue	red	blue
$p(blue)$	**1/6**	**1/3**	**1/3**	**1/2**

- $p(blue) \in [1/6,\ 1/2]$; this interval *includes* the real value.

Conservative Inference Rule

- MAR missing data are ignored.
- Non-MAR missing data are filled in all possible ways, both in the training and in the test data.
- The replacements exponentially grow with the missing data; yet polynomial time algorithms are available for NCC.

Conservative treatment of missing data increases indeterminacy

- Multiple classes are returned if the most probable class depends:
  - on the prior specification *or*
  - on the completion of the non-MAR missing data.
- Declare each feature as MAR or non-MAR depending on domain knowledge, for a good trade-off between robustness and informativeness.

Missing data with NCC

- Let us assume that there is no missing values in the class variable.
- If MAR, then a missing value for a feature is ignored in any computation involving it.
- If applying the conservative treatment, then we can efficiently compute the upper and lower probabilities w.r.t. all the possible completions of the missing values: the extreme scenarios happen when all missing values are set in favour or against the observed value of that feature in the testing instance.

Missing data with NCC

- Let us assume that there is no missing values in the class variable.
- If MAR, then a missing value for a feature is ignored in any computation involving it.
- If applying the conservative treatment, then we can efficiently compute the upper and lower probabilities w.r.t. all the possible completions of the missing values: the extreme scenarios happen when all missing values are set in favour or against the observed value of that feature in the testing instance.
- Who is up to changing some code? (If you guessed it is `classification.r` again, you might be right!)

Learning NCC with missing values

- If we assume that we are using a local IDM to learn the model, then:

$$\underline{p}(c') \cdot \prod_i \underline{p}(f_i|c') = \frac{n(c')}{n+s} \prod_i \frac{n(f_i, c')}{n(c')+s}$$

$$\overline{p}(c'') \cdot \prod_i \overline{p}(f_i|c'') = \frac{n(c'')+s}{n+s} \prod_i \frac{\overline{n}(f_i, c'')+s}{n(c'')+s}$$

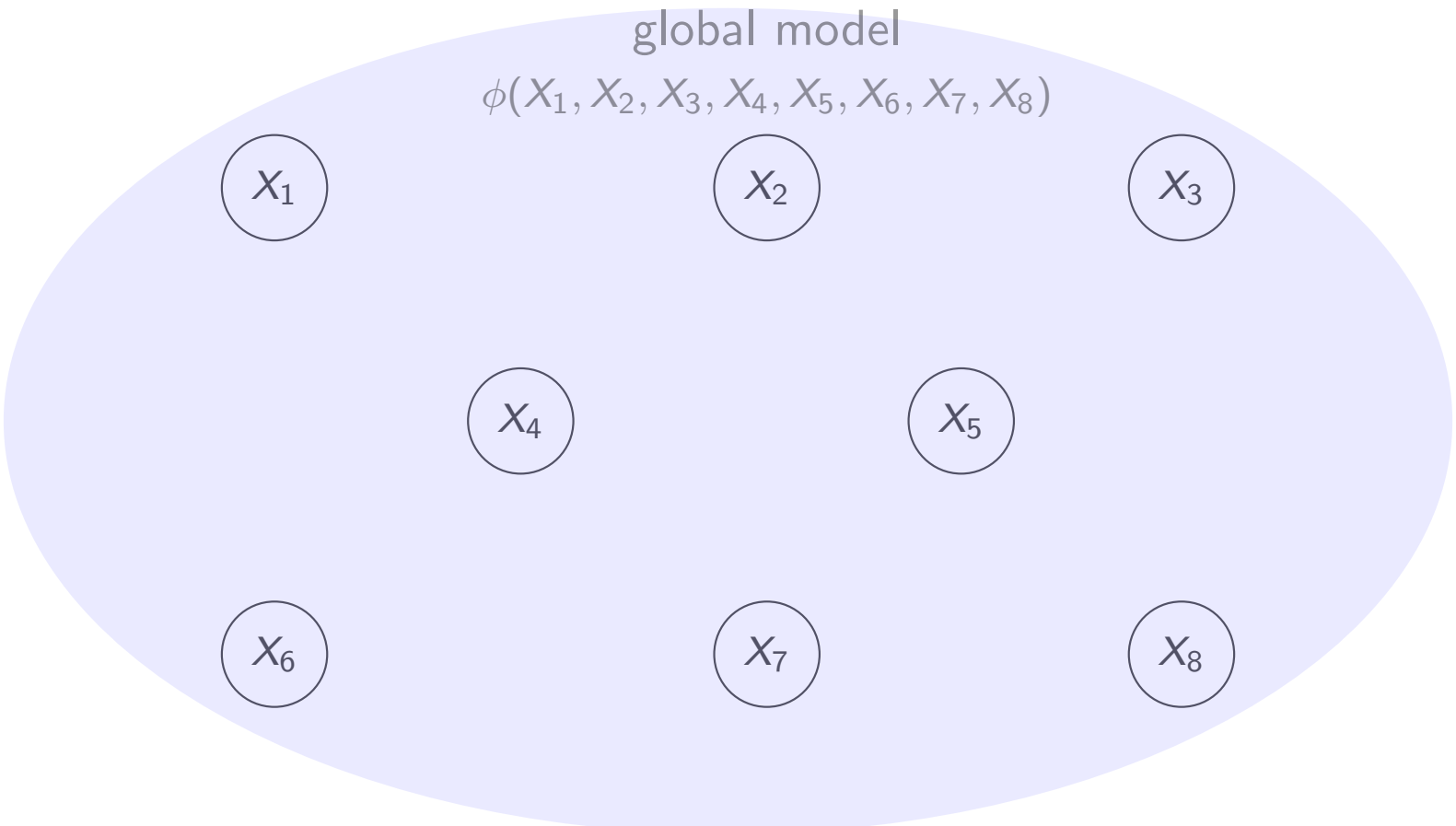
where $\overline{n}(f_i, c'') = n(f_i, c'') + n(F_i = \mathrm{NA}, c'')$.

Plan

Probabilistic Graphical Models

aka Decomposable Multivariate Probabilistic Models

(whose decomposability is induced by independence)

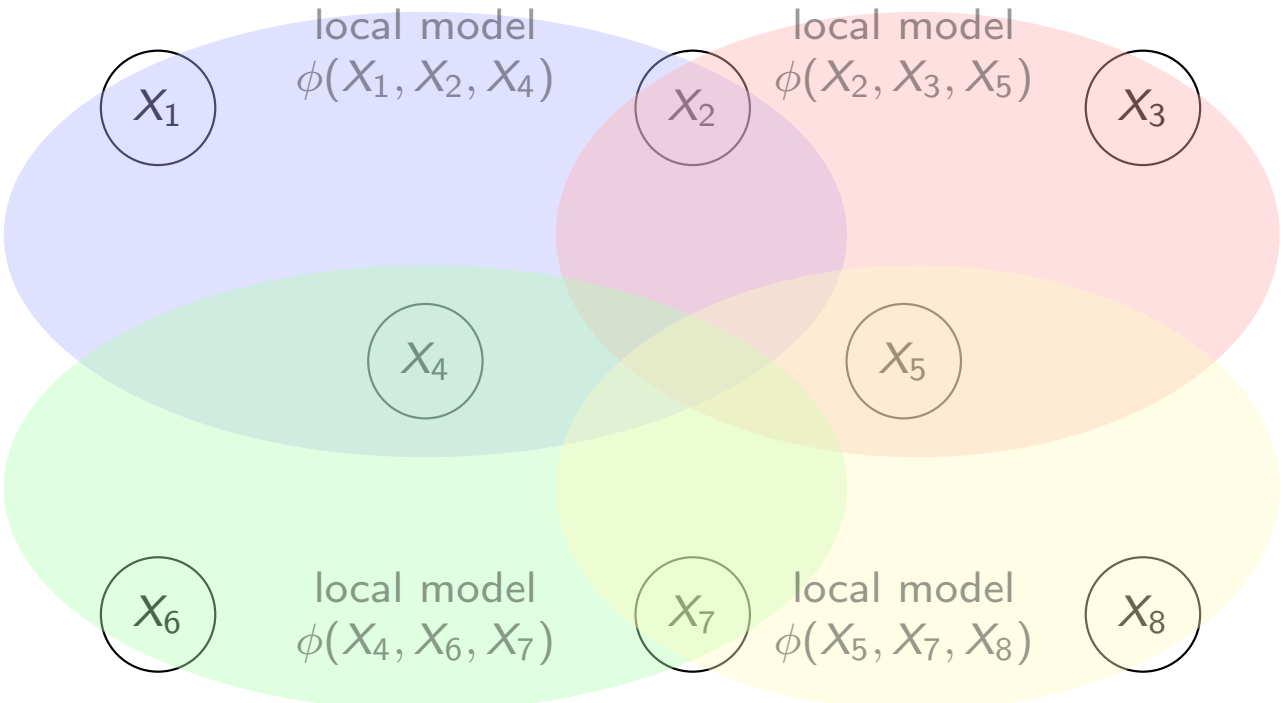


Probabilistic Graphical Models

aka Decomposable Multivariate Probabilistic Models

(whose decomposability is induced by independence)

$$\phi(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = \phi(X_1, X_2, X_4) \otimes \phi(X_2, X_3, X_5) \otimes \phi(X_4, X_6, X_7) \otimes \phi(X_5, X_7, X_8)$$



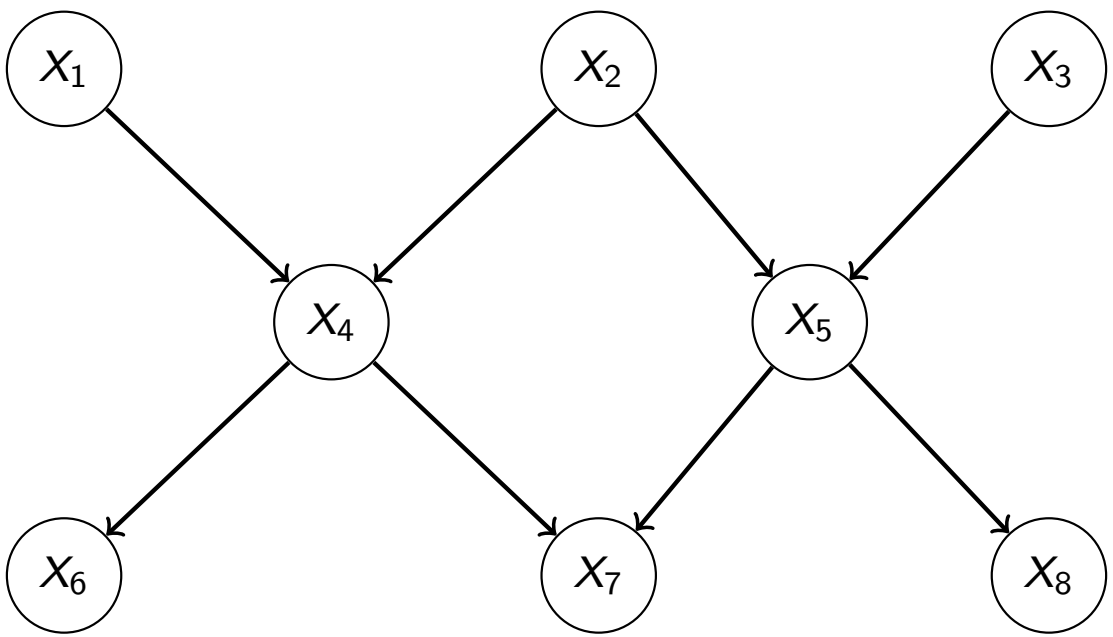
Probabilistic Graphical Models

aka Decomposable Multivariate Probabilistic Models

(whose decomposability is induced by independence)

directed graphs

Bayesian/credal networks

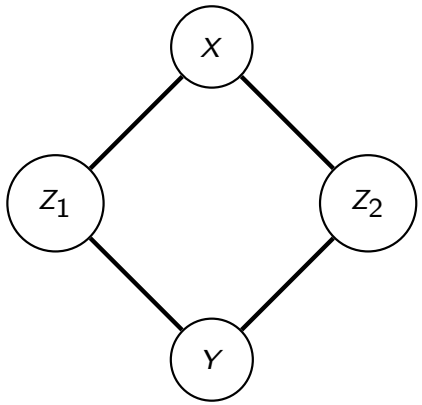


Markov Condition

- Probabilistic model over set of variables $(X_1, \ldots, X_n)$ in one-to-one correspondence with the nodes of a graph

Undirected Graphs

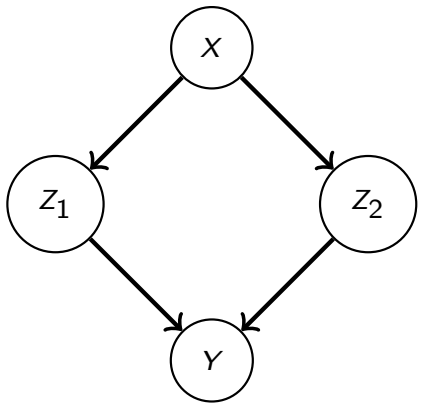
X and Y are independent given Z if any path between X and Y containts an element of Z



Directed Graphs

Given its parents, every node is independent of its non-descendants non-parents

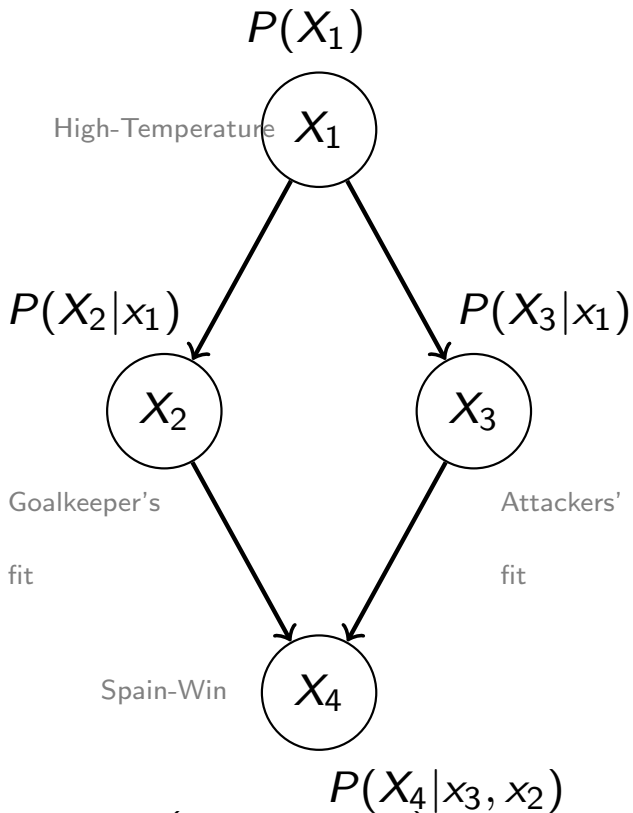
X and Y are d-separated by Z if, along every path between X and Y there is a W such that either W has converging arrows and is not in Z and none of its descendants are in Z, or W has no converging arrows and is in Z



Bayesian networks

- Set of categorical variables $X_1, \ldots, X_n$
- Directed acyclic graph
  - conditional (stochastic) independencies according to the Markov condition:

    "any node is conditionally independent of its non-descendents given its parents"
- A conditional mass function for each node and each possible value of the parents
  - $\{P(X_i|\mathrm{pa}(X_i))\ , \forall i = 1, \ldots, n\ , \forall \mathrm{pa}(X_i)\ \}$
- Defines a joint probability mass function
  - $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\mathrm{pa}(X_i))$



$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_3, x_2)$$

E.g., given temperature, fitnesses independent

Bayesian 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"))
```

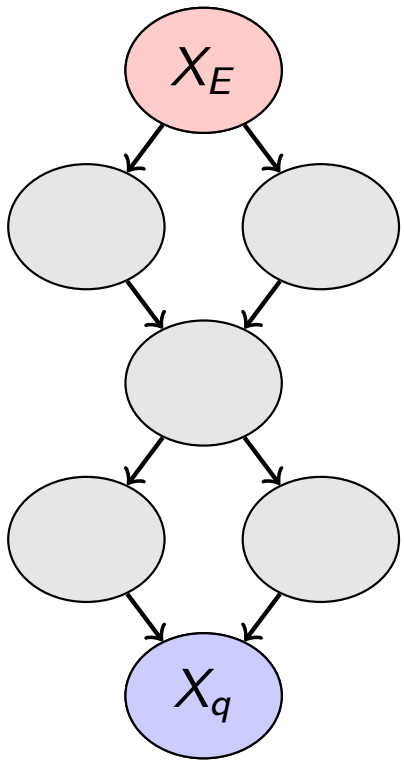
Bayesian network - simple example - bn2.txt

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

query=rep(NA,length(net.1))
names(query) <- names(net.1)
query[2]='false'
res <- my.bn.inference(net.1,query)

query[1]='true'
res <- my.bn.inference(net.1,query)

query[2]=NA
res <- my.bn.inference(net.1,query)

evidence=rep(NA,length(net.1))
names(evidence) <- names(net.1)
evidence[4]='true'
res <- my.bn.inference(net.1,query,evidence)

evidence[4]='false'
res <- my.bn.inference(net.1,query,evidence)
```

Updating Bayesian 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 polytrees)

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



$P(x_q|x_E) = .38$

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 (We will talk about credal nets with strong independence, because it has been around for more time, so we have more applications for it.)
- Convex set of joint mass functions

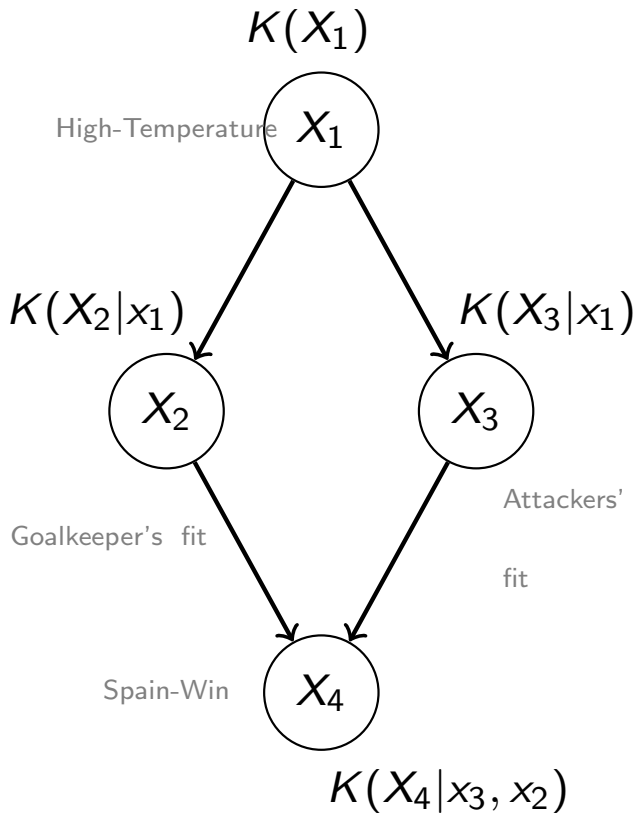
  $K(X_1,\ldots,X_n) = \mathrm{CH}\Big\{P(X_1,\ldots,X_n)\Big\}$

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

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

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

  CN $\equiv$ (exponential) number of BNs



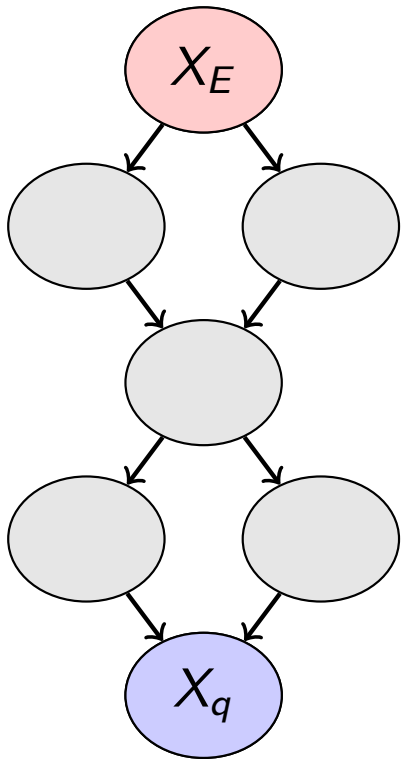
Credal network - simple example - cn1.txt

```
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.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
```

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 polytrees)

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



- Updating credal nets is $\mathrm{NP}^{\mathrm{PP}}$-hard,
  NP-hard on polytrees
  Easy in trees under epistemic irrelevance

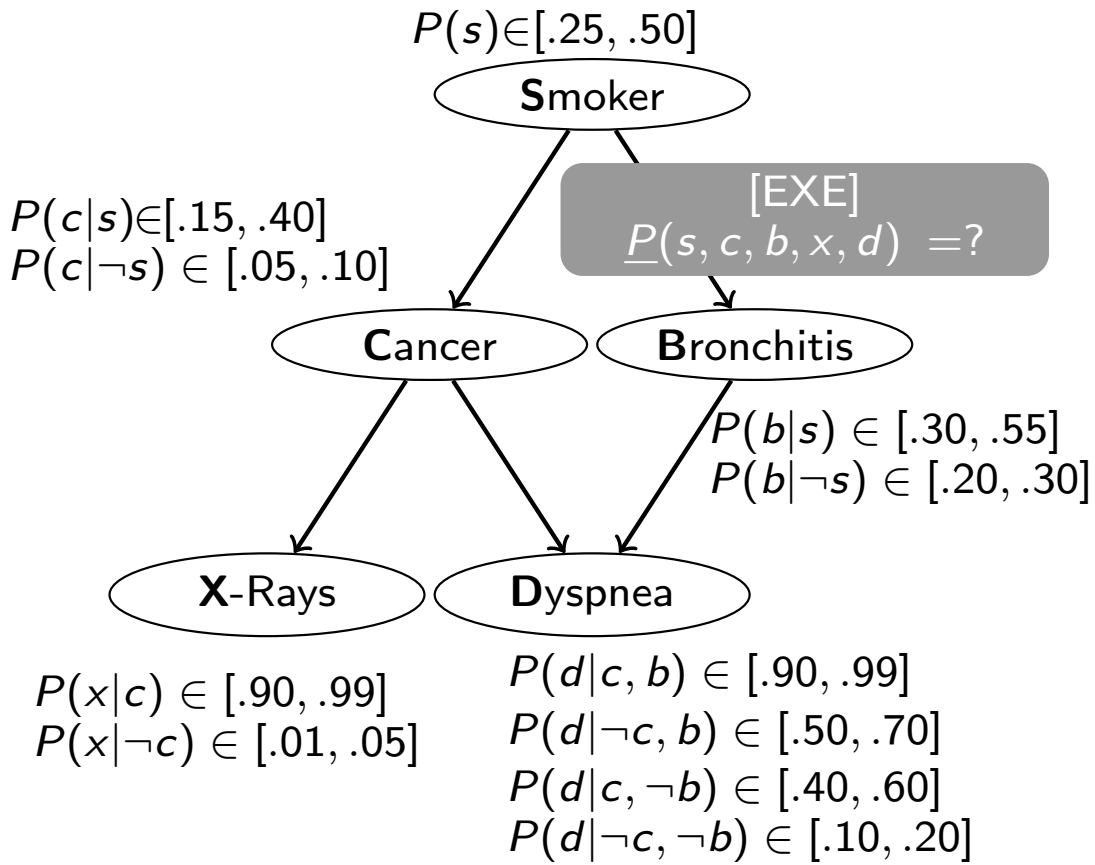
$$\underline{P}(x_q|x_E) = \min_{\substack{P(X_i|\pi_i)\in K(X_i|\pi_i)\\ i=1,\ldots,n}} \frac{\sum_{\mathbf{x}\setminus\{x_Q,x_E\}} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{\mathbf{x}\setminus\{x_Q\}} \prod_{i=1}^{n} P(x_i|\pi_i)}$$

Credal network - simple updating example - cn2.txt

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

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

query=rep(NA,length(net.1))
names(query) <- names(net.1)
query[2]='false'
res <- my.cn.inference(net.1,net.2,query)

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')
```

Medical diagnosis by CNs (a simple example of)

- 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')
```

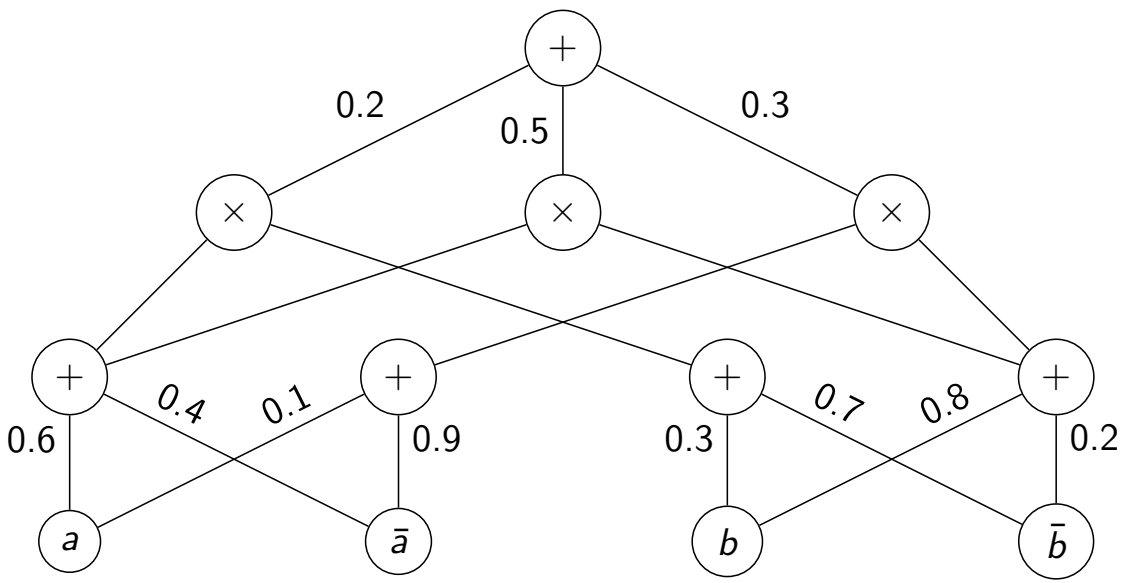
Asia network example - simpleasia4.txt

```
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

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 $S(X_1, \ldots, X_n)$ 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
  - $S_3(X, Y) = 0.6 \cdot S_1(X, Y) + 0.4 \cdot S_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $S_3(X, Y, Z, W) = S_1(X, Y) \cdot S_2(Z, W)$.

Sum-Product Network

Distribution $S(X_1, \ldots, X_n)$ 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
  - $S_3(X, Y) = 0.6 \cdot S_1(X, Y) + 0.4 \cdot S_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $S_3(X, Y, Z, W) = S_1(X, Y) \cdot S_2(Z, W)$.

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

Sum-Product Network

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

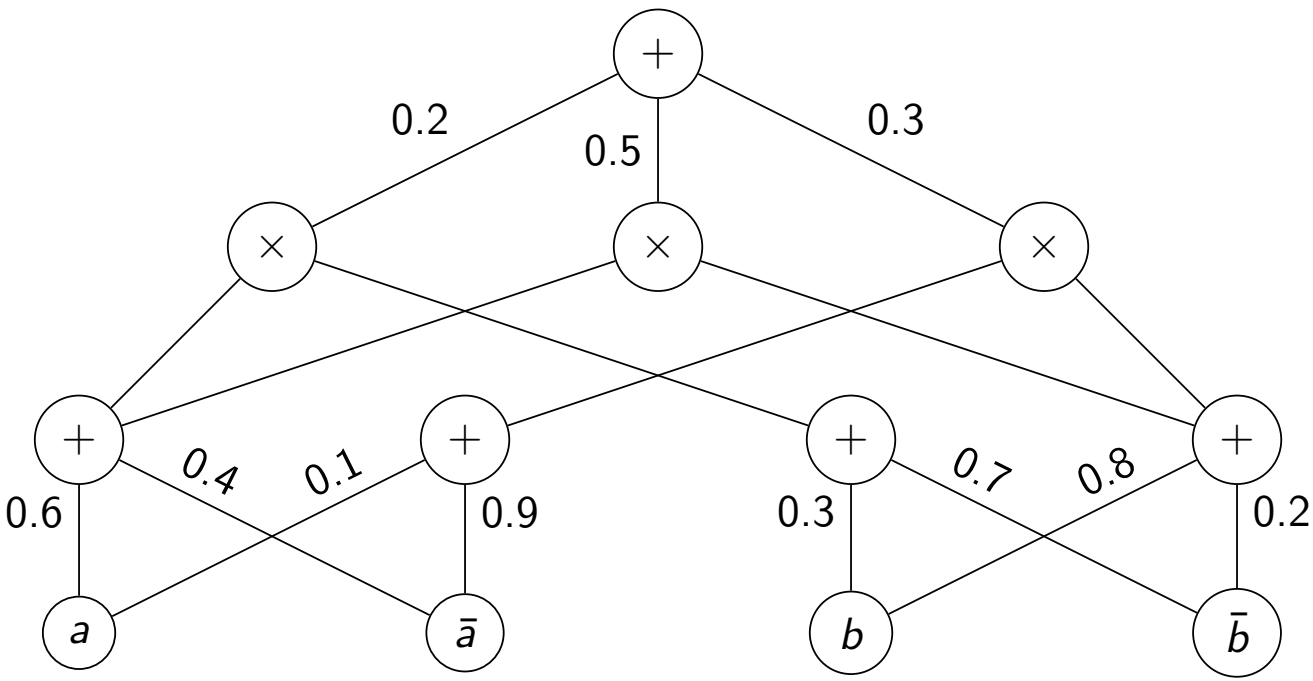
- an indicator function over a single variable
  - $I(X = 0)$, $I(Y = 1)$ (also written $\neg x, y$),
- a weighted sum of SPNs with same domain and nonnegative weights
  - $S_3(X, Y) = 0.6 \cdot S_1(X, Y) + 0.4 \cdot S_2(X, Y)$,
- a product of SPNs with disjoint domains
  - $S_3(X, Y, Z, W) = S_1(X, Y) \cdot S_2(Z, W)$.

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

Weighted sums have implicit latent variable:
$0.6 \cdot S_1(X, Y) + 0.4 \cdot S_2(X, Y) = \sum_Z P(Z) \cdot P(X, Y|Z)$.

Graphical Representation

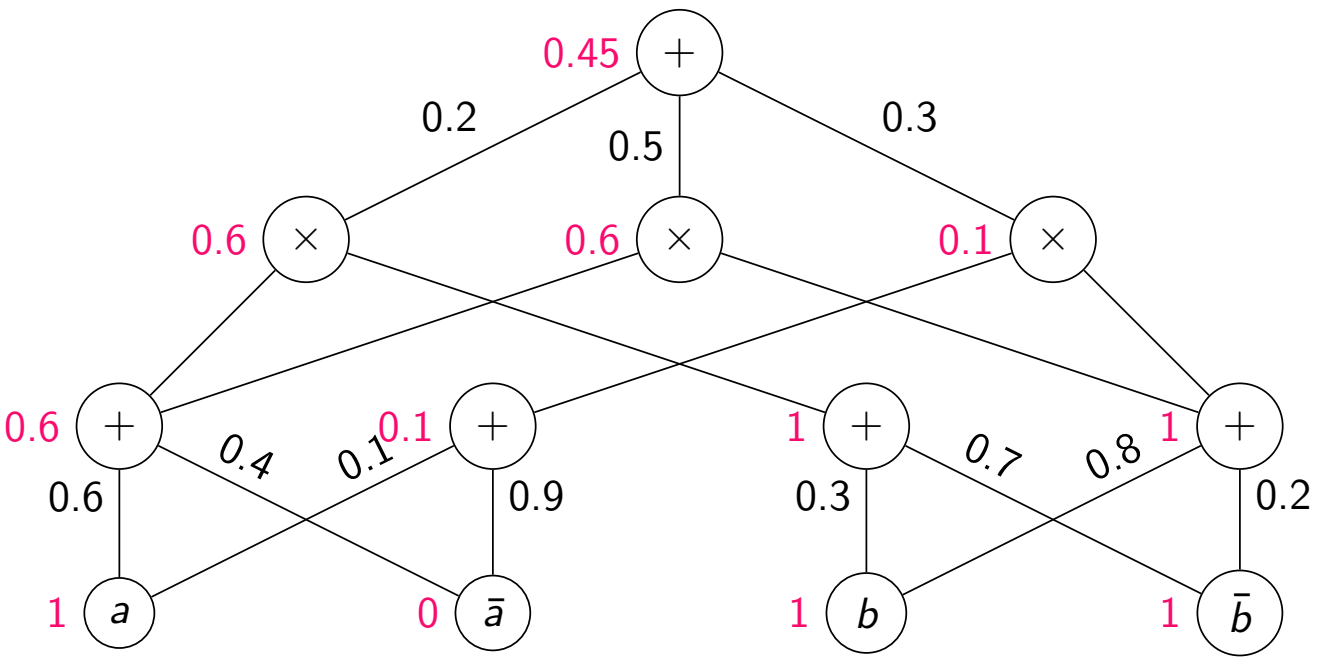
- Rooted directed acyclic graph (directions are implicit below);
- Leaves are indicators;
- Sums and product nodes (edges leaving sum nodes are weighted).



Evaluation (Inference)

- Propagate values bottom-up:

$P(A = 1) =$

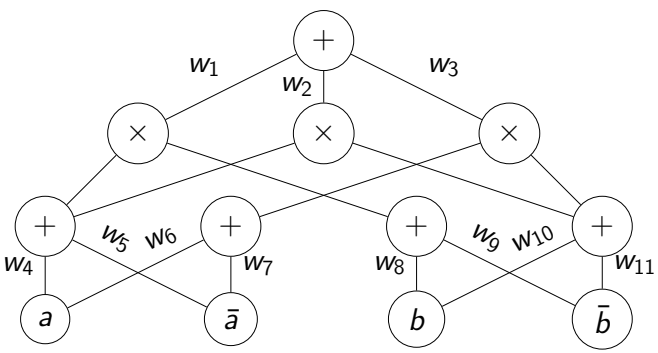


Note: takes linear time in the size of circuit!

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.

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



Credal classification

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

$$\min_{w} \left(S_w(c', e) - S_w(c'', e)\right) > 0.$$

Credal classification

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

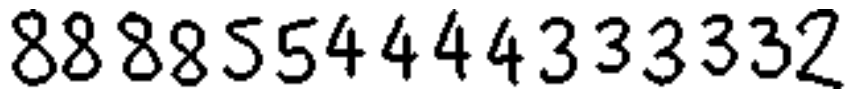
$$\min_w \left(S_w(c', e) - S_w(c'', e)\right) > 0.$$

Theorem

Credal classification with a single class variable can be done in polynomial time when each internal node has at most one parent.

Note: 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!

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!
- I feel like it is time to code again, what about you?

Plan

Motivations

- Hypothesis tests are ubiquitous. Decision making, scientific discoveries, feature selection in many fields (e.g., medical, demographic, environmental, etc.) are based on the results of hypothesis tests.
- In case of scarce prior knowledge of the distributions of interest, **nonparametric tests** are preferred (robust to outliers, do not assume normality of the data,...)
- Null hypothesis significance testing is usually adopted. However
  - Typical hypothesis tests cannot assess evidence for the null hypothesis.
  - Lack of a sound criterion for deciding Type I error (usually 0.05 or 0.01).
  - the p-value and thus the outcome of the test depend on the intention of the person who has collected the data.

Motivations

- A Bayesian nonparametric approach evaluates the posterior probability of the alternative hypothesis.
- This allows taking decisions which minimize the expected loss once the costs of type I and type II errors are specified.
- The outcome of the test depends only on the prior belief and on the collected data.

The Dirichlet Process

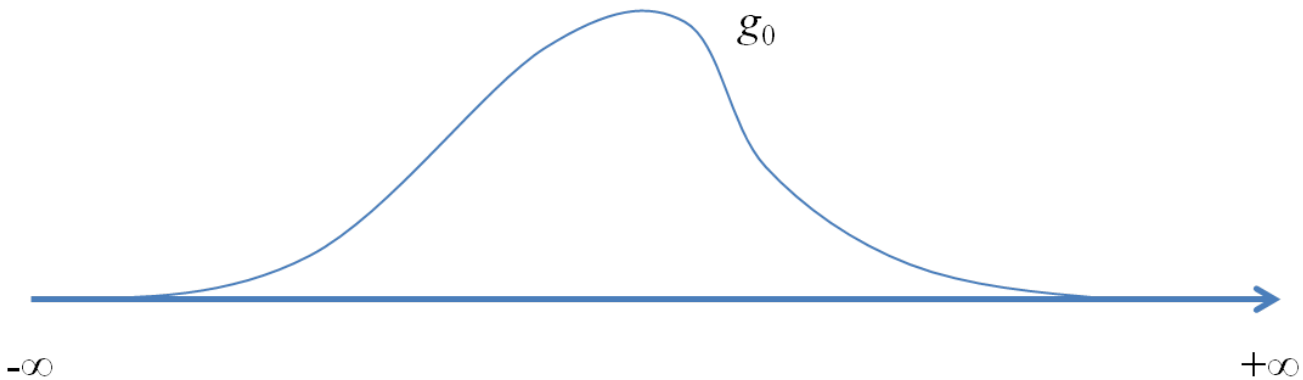
- The Dirichlet process (DP) is one of the most popular Bayesian nonparametric models.
- In Ferguson's seminal paper it is used to estimate:
  - distribution function, mean, quantiles, variance;
  - $P(X < Y) \rightarrow$ **Mann-Whitney**;
  - *survival function* $\rightarrow$ **Kaplan-Meier** (Susarla, Van Ryzin and Blum);
  - measure of *bivariate dependence* $\rightarrow$ **Kendall's tau** (Dalal and Phadia);
  - ...
- DP can be used to perform traditional tests in a Bayesian way.

The Dirichlet Process - definition

- The DP is a way of assigning a probability distribution over probability distributions.
- Let the probability measure P be distributed as Dirichlet process $\mathrm{Dp}(s, g_0)$:
  - $s =$ prior strength (scalar);
  - $g_0 =$ prior base probability measure on Ω (infinite dimensions).
- Then, given a finite partition $B_1, B_2, \ldots, B_m$ of Ω, $(P(B_1), \ldots, P(B_m)) \sim \mathrm{Dir}(s\ g_0(B_1), \ldots, s\ g_0(B_m))$.

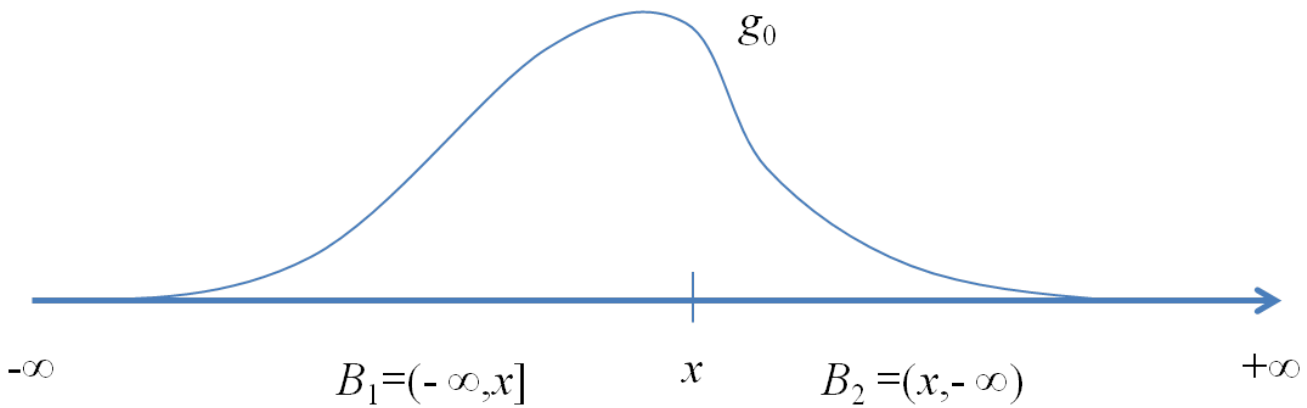
The Dirichlet Process on $\mathbb{R}$

- Sample space $\Omega = \mathbb{R}$;
- $P \sim \mathrm{Dp}(s, g_0)$;



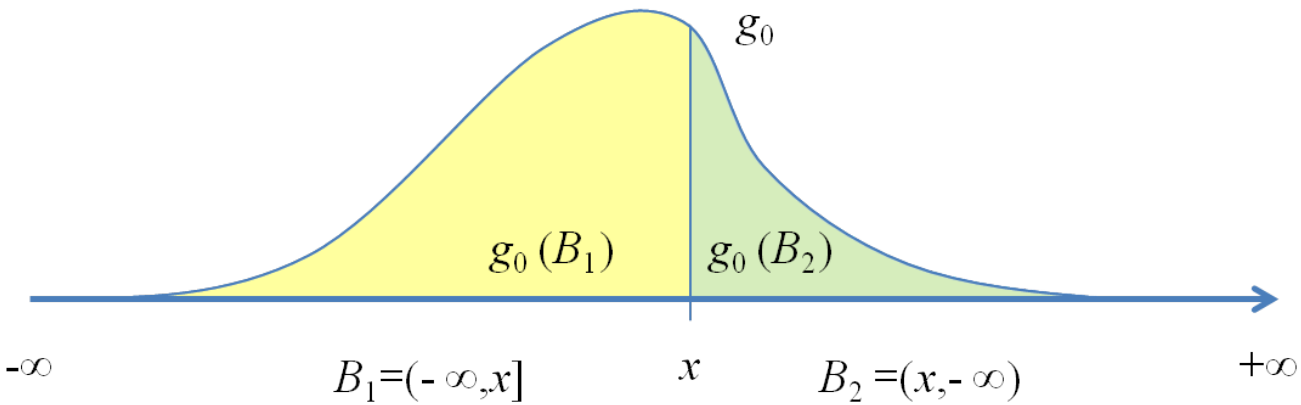
The Dirichlet Process on $\mathbb{R}$

- Sample space $\Omega = \mathbb{R}$;
- $P \sim \mathrm{Dp}(s, g_0)$;



The Dirichlet Process on $\mathbb{R}$

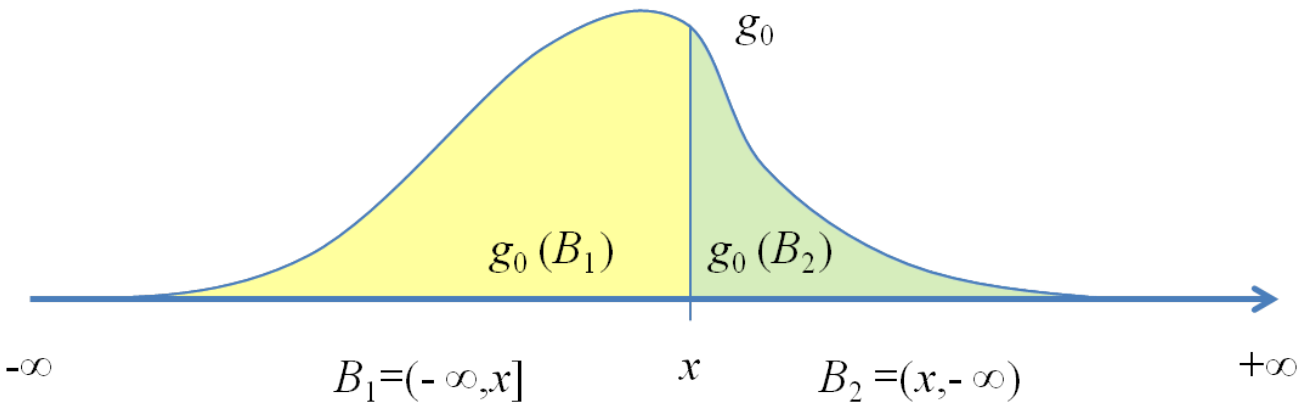
- Sample space $\Omega = \mathbb{R}$;
- $P \sim \mathrm{Dp}(s, g_0)$;



$$P(X < x) = \quad P(B_1) \sim \mathrm{Dir}(sg_0(B_1), sg_0(B_2))$$
$$\sim \mathrm{Beta}(sg_0(B_1), s[1 - g_0(B_1)]) \implies \mathcal{E}[P(X < x)] = g_0(B_1)$$

The Dirichlet Process on $\mathbb{R}$

- Sample space $\Omega = \mathbb{R}$;
- $P \sim \mathrm{Dp}(s, g_0)$;

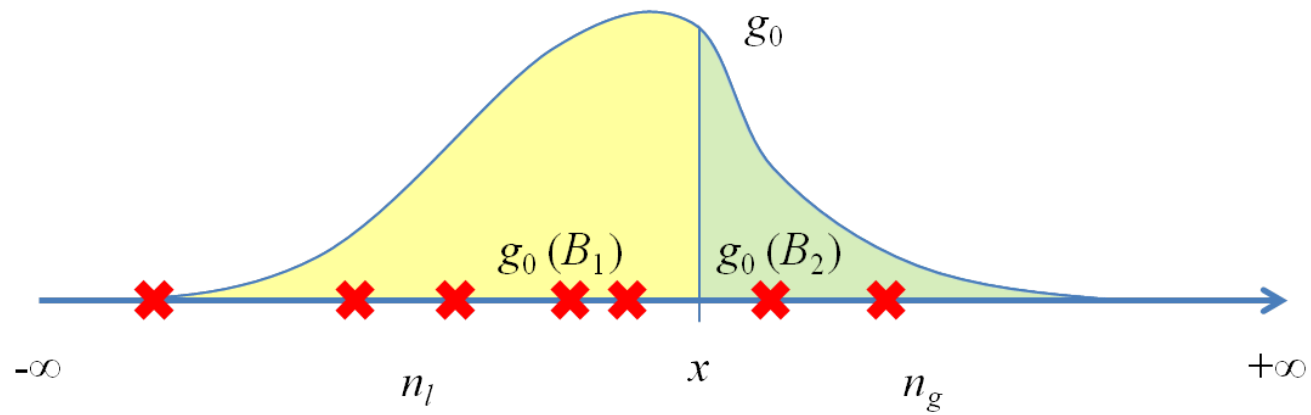


$$P(X < x) = \quad P(B_1) \sim \mathrm{Dir}(sg_0(B_1), sg_0(B_2))$$
$$\sim \mathrm{Beta}(sg_0(B_1), s[1 - g_0(B_1)]) \implies \mathcal{E}[P(X < x)] = g_0(B_1)$$

g_0 represents our prior belief about the shape of P.

The Dirichlet Process on $\mathbb{R}$

Let $X_1, X_2, \ldots$ be n samples from P



Prior: $P(X < x) \sim \text{Dir}(sg_0(B_1), sg_0(B_2))$

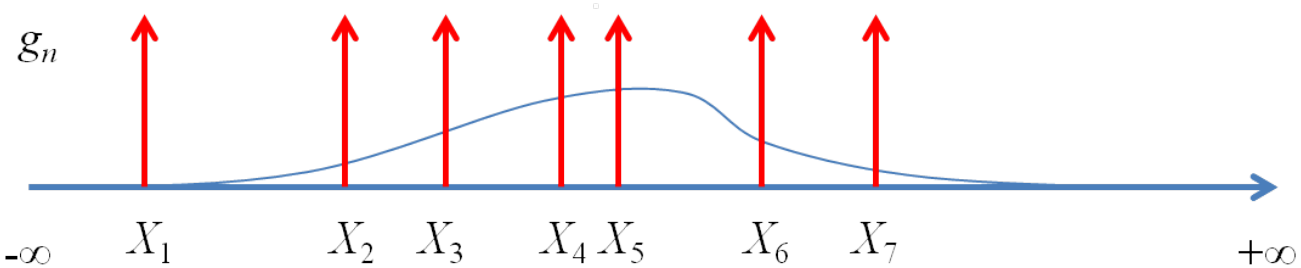
Posterior: $P(X < x) \sim \text{Dir}(sg_0(B_1) + n_l, sg_0(B_2) + n_g)$

$$\implies \mathcal{E}[P(X < x)] = \frac{sg_0(B_1) + n_l}{s + n}$$

The Dirichlet Process on $\mathbb{R}$ - Conjugacy

Prior: $P \sim \mathrm{Dp}(s, g_0)$

Posterior: $P \sim \mathrm{Dp}(s+n, \underbrace{\frac{s}{s+n} g_0 + \frac{1}{n+s} \sum_{i=1}^{n} \delta_{X_i}}_{g_n})$



Prior elicitation

In Bayesian analysis the problem is how to select s and g_0.

Note that g_0 is a probability measure and, thus, very detailed information is needed for its elicitation.

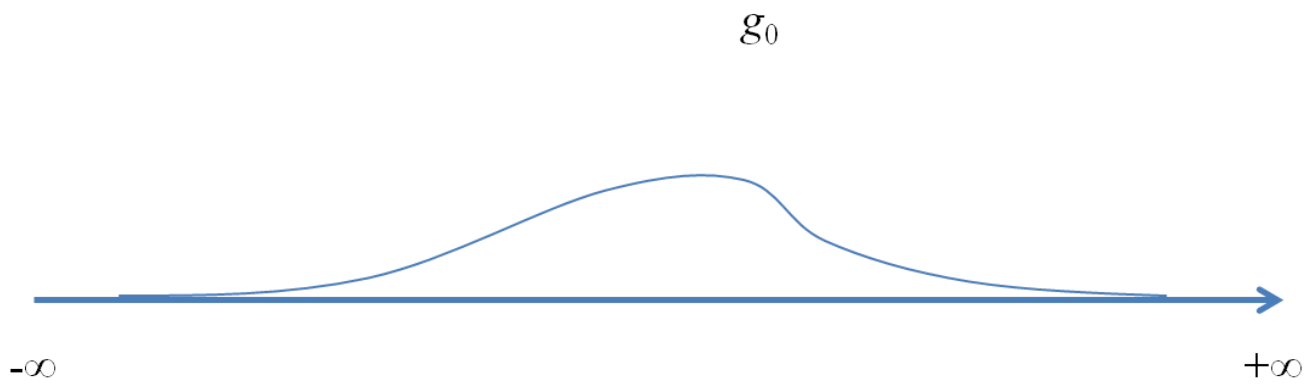
In case of lack of prior information:

- $s \to 0$ (Ferguson, Rubin);
- s, g_0 are selected using empirical Bayesian approaches;
- hierarchical prior on s, g_0.

Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

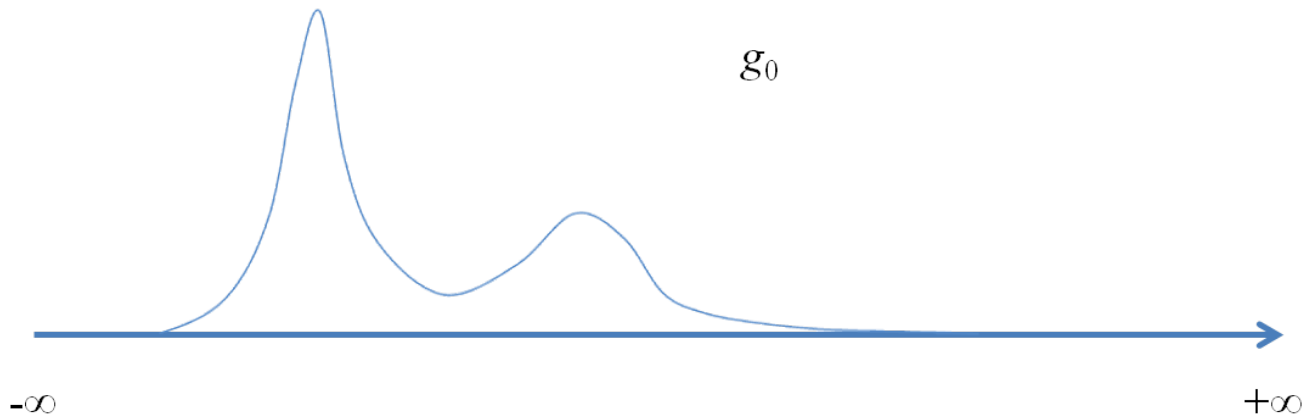
$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$



Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

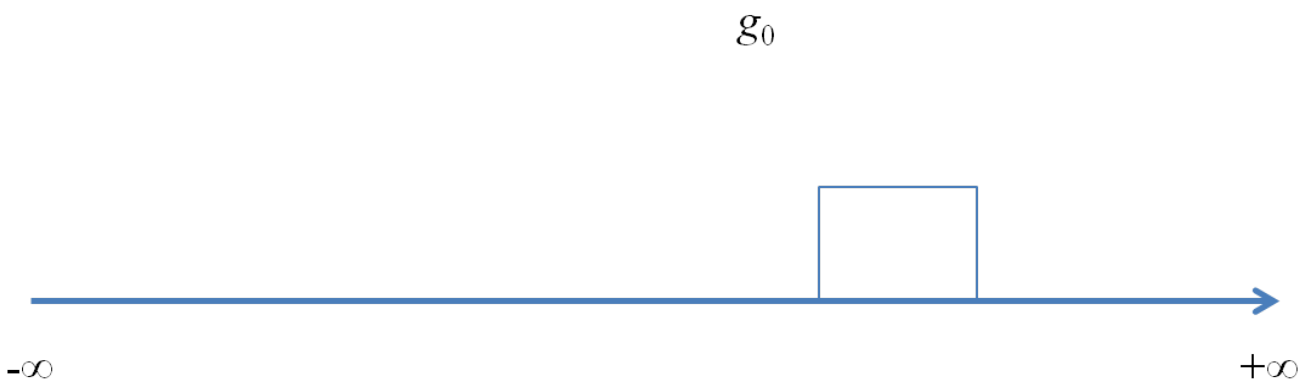
$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$



Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

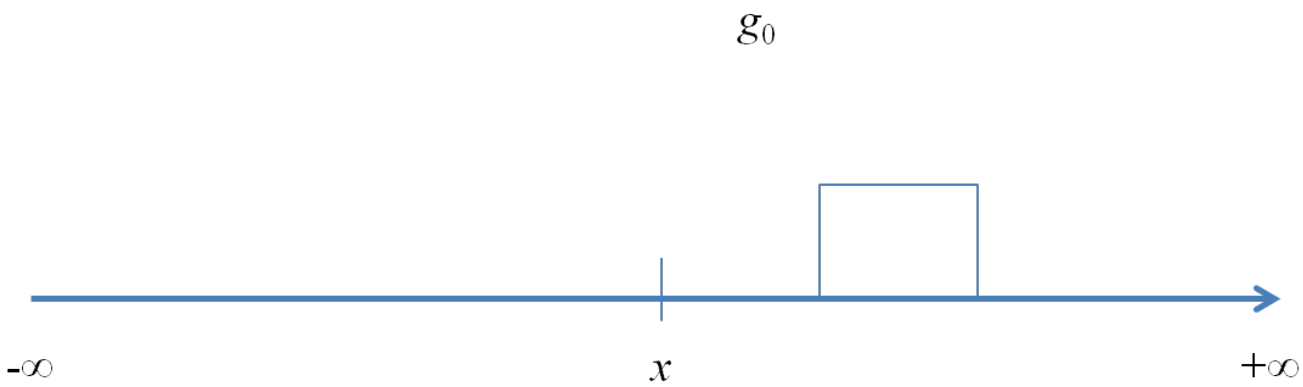
$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$



Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$

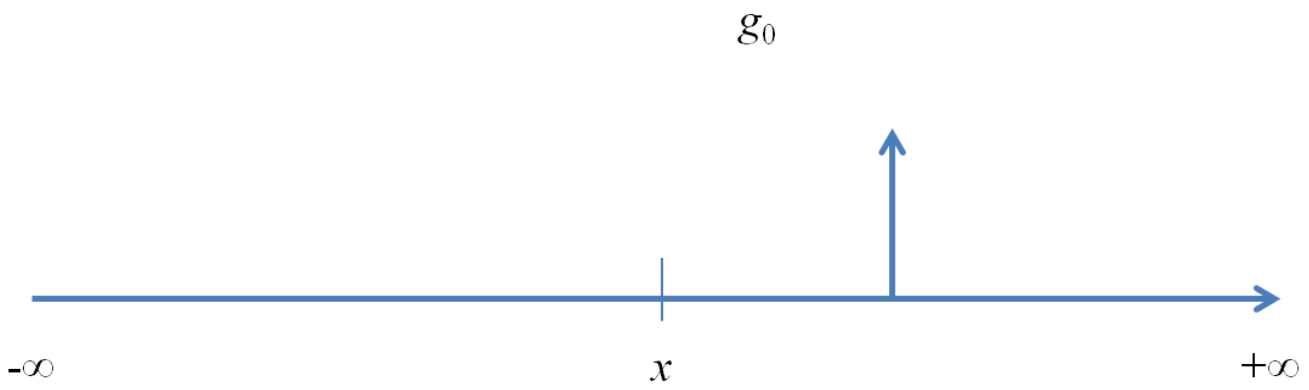


$$\mathcal{E}[P(X < x)] = g_0(-\infty, x] = 0$$

Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$

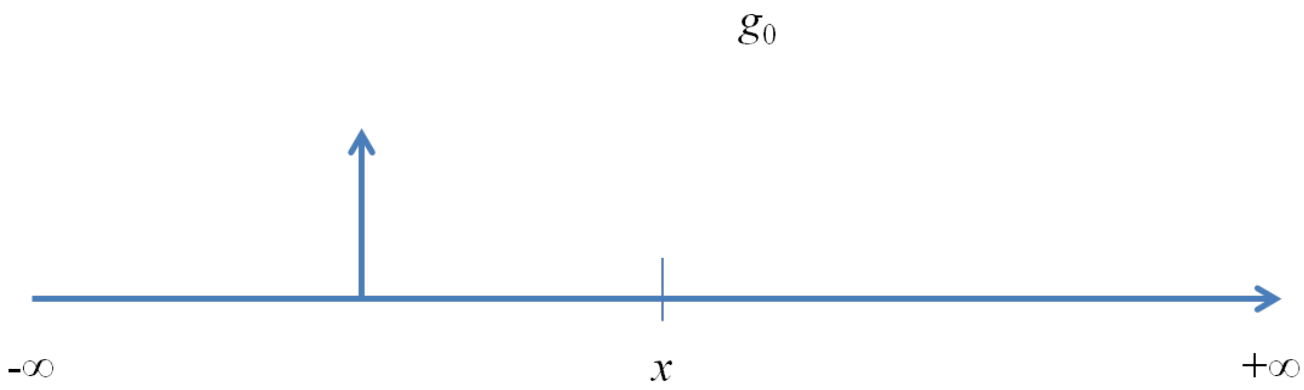


$$\mathcal{E}[P(X < x)] = g_0(-\infty, x] = 0$$

Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$

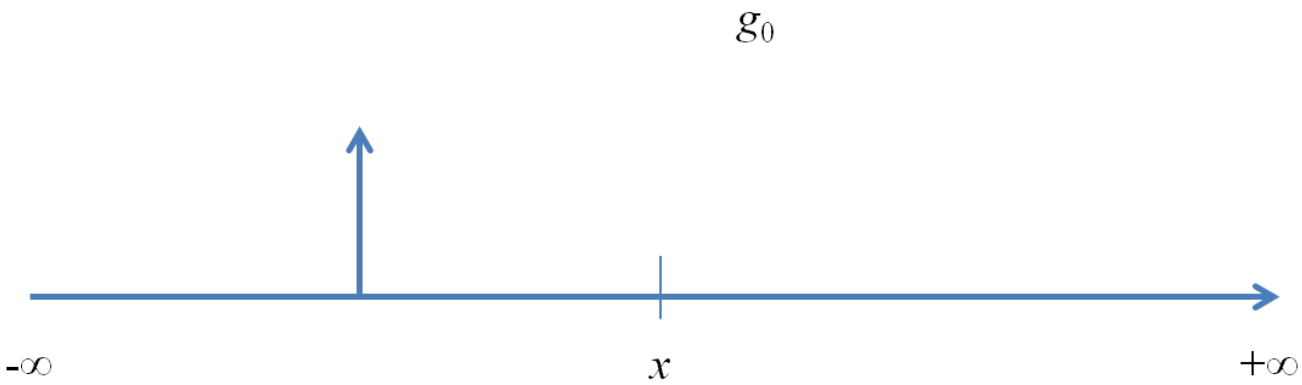


$$\mathcal{E}[P(X < x)] = g_0(-\infty, x] = 1$$

Near-ignorance solution: Imprecise Dirichlet Process

Keep s fixed and let g_0 vary in the set of all probability measures:

$$IDP: \quad \{Dp(s, g_0), \quad g_0 \in \mathbb{P}\}$$



$$\mathcal{E}[P(X < x)] = g_0(-\infty, x] = 0 \qquad \mathcal{E}[P(X < x)] = g_0(-\infty, x] = 1$$

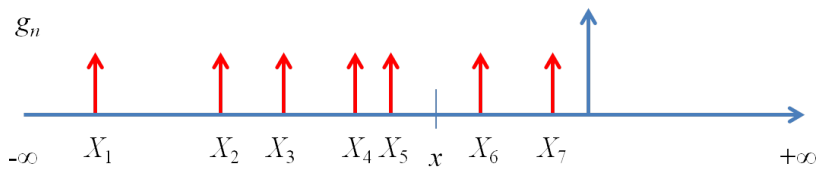
$$0 \leq \mathcal{E}[P(X < x)] \leq 1$$

No prior information about $P(X < x)$

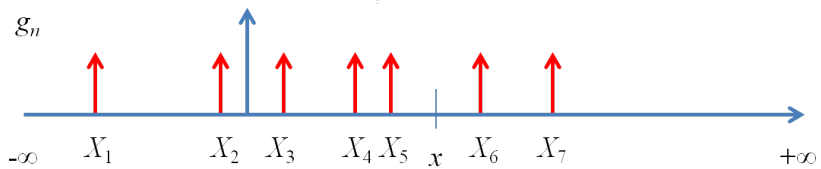
Imprecise Dirichlet Process - Learning

A posteriori:

LOWER

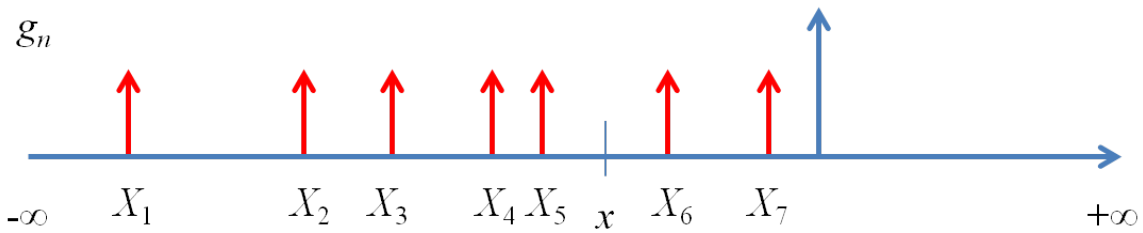


UPPER



$$\frac{n_l}{s+n} < \mathcal{E}[P(X < x)] < \frac{s+n_l}{s+n}$$

Imprecise Dirichlet Process - Sampling

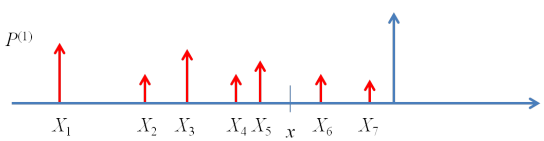


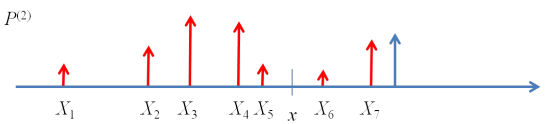
$$g_0 = \delta_{X_0} \Rightarrow g_n \text{ is discrete}$$

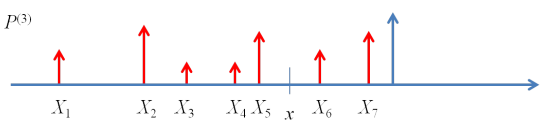
Samples $P^{(k)}$ from $Dp(s+n, g_n)$ have the form

$$P^{(k)} = w_0 \delta_{X_0} + \sum_{i=1}^{n} w_i \delta_{X_i}$$

$$\text{with } (w_0, w_1, \ldots, w_n) \sim \text{Dir}(s, \overbrace{1, \ldots, 1}^{n})$$







Sign test

$X^n = \{X_1, \ldots, X_n\}$: sequence of observations

n_l: # observations $X_i < 0$ $\quad$ n_g: # observations $X_i \geq 0$

$$H_0 : \ P(X < 0) \leq 0.5 \quad H_1 : \ P(X < 0) > 0.5$$

Lower

$P(X < 0) \sim \text{Beta}(n_l, s + n_g)$

$\underline{\text{Prob}}(H_1) = \int_{0.5}^{1} \text{Beta}(n_l, s + n_g)$

Upper

$P(X < 0) \sim \text{Beta}(s + n_l, n_g)$

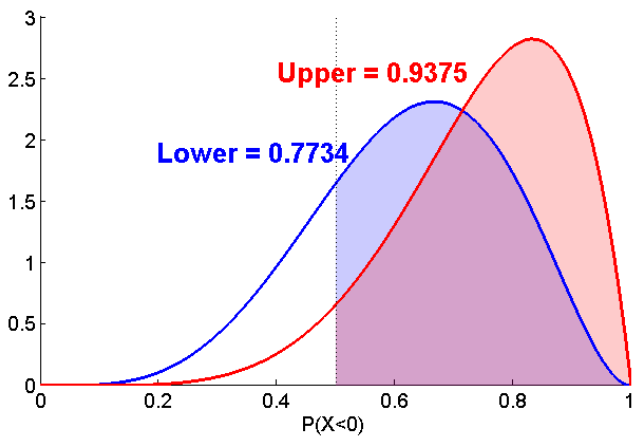
$\overline{\text{Prob}}(H_1) = \int_{0.5}^{1} \text{Beta}(s + n_l, n_g)$

Example:

$n_l = 5,$

$n_g = 2,$

$s = 1$



The sign test - Decision making

L_0: loss associated with action a_1 (= accepting H_1) if H_0 is true (Type I error)

L_1: loss associated with action a_0 (= accepting H_0) if H_1 is true (Type II error)

Based on L_0, L_1 one chooses a_1 (accept H_1) if

$$\begin{aligned} &\text{Expected loss}|a_1 < \text{Expected loss}|a_0 \\ &\Rightarrow L_0 \text{Prob}(H_0) < L_1 \text{Prob}(H_1) \\ &\Rightarrow \text{Prob}(H_1) > \frac{L_0}{L_0 + L_1} = 1 - \alpha. \end{aligned}$$

What if $\overline{\text{Prob}}(H_1) > 1 - \alpha$ but $\underline{\text{Prob}}(H_1) < 1 - \alpha$?

The decision is prior-dependent. No robust decision can be taken.

Advantages

Computational tractability: Sampling from the upper and lower posterior distribution is easier than using stick breaking or other sampling strategies specific to DP.

Robustness: When the IDP test is indeterminate the traditional test virtually behaves as a random guesser (50% of the times issues H_0 and the other 50% H_1).
The instances that are prior-dependent are somehow critical.
It makes sense to suspend the decisions in those instances.

Sensitivity analysis: The maximum value of s that gives a determinate decision can be interpreted as a measure of robustness of the decision.
Even collecting s more observations I will not contradict that decision.

The IDP statistical package

`http://ipg.idsia.ch/software/IDP.php`

(Unfortunately mostly in Matlab)

- IDP based version of the *Wilcoxon rank-sum test*;
- IDP based version of the *Wilcoxon signed-rank test*;
- IDP based version of the *sign test*;
- IDP for analysis of survival data (`IDPSurvival`).

The IDP statistical package

`http://ipg.idsia.ch/software/IDP.php`

(Unfortunately mostly in Matlab)

- IDP based version of the *Wilcoxon rank-sum test*;
- IDP based version of the *Wilcoxon signed-rank test*;
- IDP based version of the *sign test*;
- IDP for analysis of survival data (`IDPSurvival`).

Should we try to work with one of these tests now or are we all too tired? :-)

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.
- Benavoli, A., Mangili, F., Ruggeri, F., Zaffalon, M. (2015). *Imprecise Dirichlet process with application to the hypothesis test on the probability that $X \leq Y$*. Journal of Statistical Theory and Practice 9, pp. 658–684.
- 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. (2007). *Inference in credal networks through integer programming*. International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), pp. 145-154.

Some references

- 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.
- Conaty, D., de Campos, C., Martinez Del Rincon, J. (2018). *Cascading Sum-Product Networks using Robustness*. Int. Conference on Probabilistic Graphical Models (PGM), to appear in Proc. of Machine Learning Research (PMLR).
- de Cooman, G., Zaffalon, M. (2004). *Updating beliefs with incomplete observations*. Artificial Intelligence 159(12), 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.
- Ferguson, T. S. (1973). *A Bayesian Analysis of Some Nonparametric Problems*. Annals of Statistics 1(2), pp. 209–230.

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.