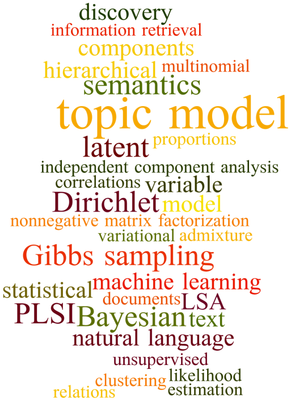
Accurate parameter estimation for **Bayesian network classifiers** using hierarchical Dirichlet processes

François Petitjean, **Wray Buntine**,
Geoff Webb and Nayyar Zaidi

Monash University

2018-09-13

Outline


discovery
information retrieval
components
hierarchical
multinomial
semantics
topic model
latent
proportions
independent component analysis
correlations
variable
Dirichlet
model
nonnegative matrix factorization
variational
admixture
Gibbs sampling
statistical
machine learning
documents
LSA
PLSI
Bayesian
text
natural language
unsupervised
clustering
likelihood
relations
estimation

A Cultural Divide

Context: When discussing teaching Data Science with a well known professor of Statistics.

She said: "when first teaching overfitting, I always give some examples where machine learning has trouble, like decision trees"

I said: "funny, I do the reverse, I always give examples where statistical models have trouble"

A Cultural Divide

Context: When discussing teaching Data Science with a well known professor of Statistics.

She said: "when first teaching overfitting, I always give some examples where machine learning has trouble, like decision trees"

I said: "funny, I do the reverse, I always give examples where statistical models have trouble"

ASIDE: our hierarchical smoothing also gives state of the art results for decision tree smoothing

State of the Art in Classification

Favoured techniques for standard classification are Random Forest and Gradient Boosting (of trees).

State of the Art in Classification

Favoured techniques for standard classification are Random Forest and Gradient Boosting (of trees).

NB. for sequences, images or graphs, deep neural networks (recurrent NN, convolutional NN, etc.) are better

Main Claim

Main Claim: Hierarchical smoothing applied to Bayesian network classifiers on categorical data beats Random Forest

[1] not well shown in the paper ...

Main Claim

Main Claim: Hierarchical smoothing applied to Bayesian network classifiers on categorical data beats Random Forest

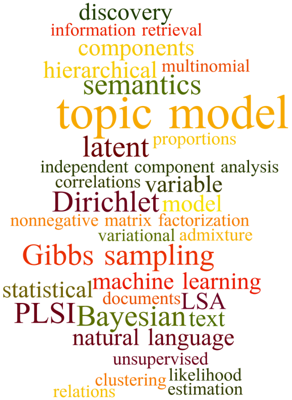
- a single model beats state of the art ensemble
  - is also comparable with XGBoost[1]
- but only on categorical data
  - though also for a lot of other data too[1]

[1] not well shown in the paper ...

Unpacking the Main Claim

- **Hierarchical smoothing**
  - using hierarchical Dirichlet models
- applied to **Bayesian network classifiers**
  - the KDB and SKDB family
- on categorical **datasets**
  - or pre-discretised attributes
- **beats Random Forest**

Outline



Motivation

Bayesian Network Classifiers

Hierarchical Smoothing

Experimental Setup

Results

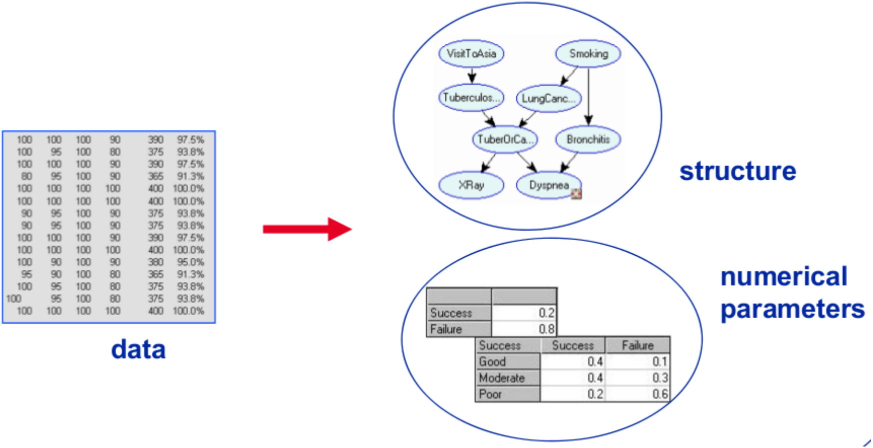
Conclusion

Reminder: Main Claim

- Hierarchical smoothing
- applied to **Bayesian network classifiers**
  - the KDB and SKDB family
- on categorical datasets
- beats Random Forest

Learning Bayesian Networks

tutorial by Cussens, Malone and Yuan, *IJCAI* 2013



Bayesian Networks learning =

Structure learning + Conditional Probability Table estimation

Bayesian Network Classifiers

Friedman, Geiger, Goldszmidt, *Machine Learning* 1997

- Defined by parent relation π and Conditional Probability Tables (CPTs)
  - π encodes conditional independence / structure
  - π_i is the parent variables for X_i
  - CPTs encode conditional probabilities
- For classification, make class variable Y a parent of all X_i

Bayesian Network Classifiers

Friedman, Geiger, Goldszmidt, *Machine Learning* 1997

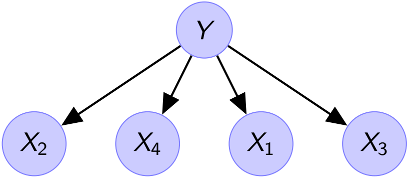
- Defined by parent relation π and Conditional Probability Tables (CPTs)
  - π encodes conditional independence / structure
  - π_i is the parent variables for X_i
  - CPTs encode conditional probabilities
- For classification, make class variable Y a parent of all X_i
- Classifies using $\mathrm{P}(y \mid \mathbf{x}) \propto \mathrm{P}(y \mid \pi_Y) \prod \mathrm{P}(x_i \mid \pi_i)$

Bayesian Network Classifiers

Friedman, Geiger, Goldszmidt, *Machine Learning* 1997

- Defined by parent relation π and Conditional Probability Tables (CPTs)
  - π encodes conditional independence / structure
  - π_i is the parent variables for X_i
  - CPTs encode conditional probabilities
- For classification, make class variable Y a parent of all X_i
- Classifies using $\mathrm{P}(y \mid \mathbf{x}) \propto \mathrm{P}(y \mid \pi_Y) \prod \mathrm{P}(x_i \mid \pi_i)$

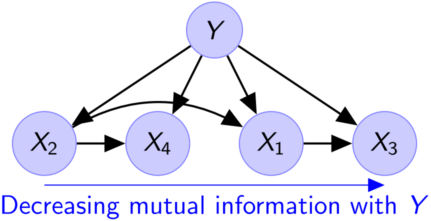
Naïve Bayes classifier:
$\pi_i = \{Y\}$



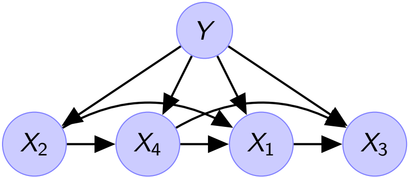
k-Dependence Bayes (KDB)

Sahami, *KDD* 1996

KDB-1 classifier:
(attributes have 1 extra parent)



KDB-2 classifier:
(attributes have 2 extra parents)



NB. other parents also selected by mutual information

Learning k-Dependence Bayes (KDB)

- **Two pass learning**
- 1st pass, learn structure π:
  - Uses variable ordering heuristics based on mutual information, so efficient and scalable.

Learning k-Dependence Bayes (KDB)

- **Two pass learning**
- 1st pass, learn structure π:
  - Uses variable ordering heuristics based on mutual information, so efficient and scalable.
- 2nd pass, learn CPTs:
  - Collect statistics according to the structure learned.
  - Form CPTs using Laplace smoothers, or m-estimation.
  - With simple CPTs is exponential family so inherently scalable.

Selective k-Dependence Bayes (SKDB)

Martnez, Webb, Chen and Zaidi, *JMLR* 2016

But, how do we pick k in KDB, and how do we select which attributes to use?

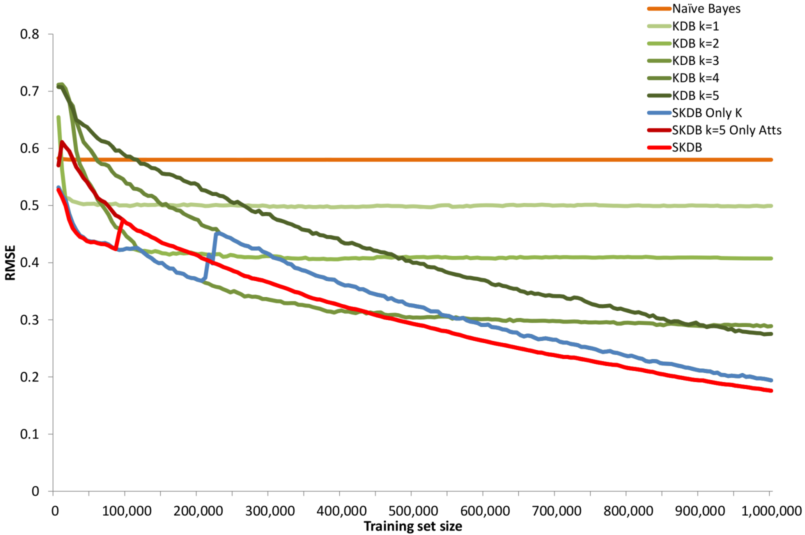
Selective k-Dependence Bayes (SKDB)

Martnez, Webb, Chen and Zaidi, *JMLR* 2016

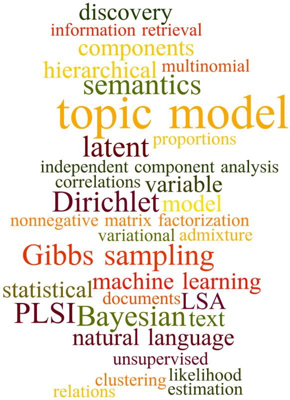
But, how do we pick k in KDB, and how do we select which attributes to use?

- Use Leave-one-out cross validation (LOOCV) on MSE to select both k and which attributes to use.
- Requires a third pass through the data to compute LOOCV MSE estimates of probability and minimise.
- As efficient as previous passes.
- Called SKDB.

Learning Curves: Typical Comparison



Outline

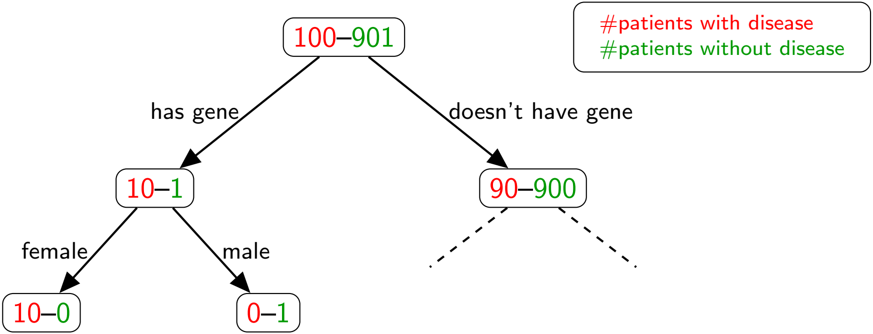

discovery
information retrieval
components
hierarchical
multinomial
semantics
topic model
latent
proportions
independent component analysis
correlations
variable
Dirichlet
model
nonnegative matrix factorization
variational
admixture
Gibbs sampling
statistical
machine learning
documents
LSA
PLSI
Bayesian
text
natural language
unsupervised
clustering
likelihood
relations
estimation

Reminder: Main Claim

- **Hierarchical smoothing**
  - using hierarchical Dirichlet models
- applied to Bayesian network classifiers
- on categorical datasets
- beats Random Forest

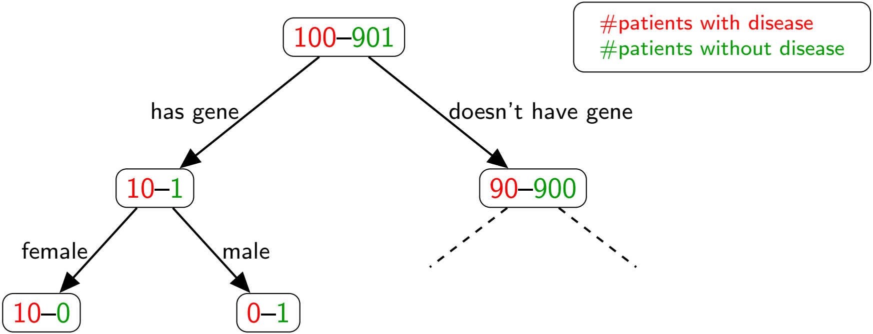
Why doing Hierarchical Smoothing?

- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



Why doing Hierarchical Smoothing?

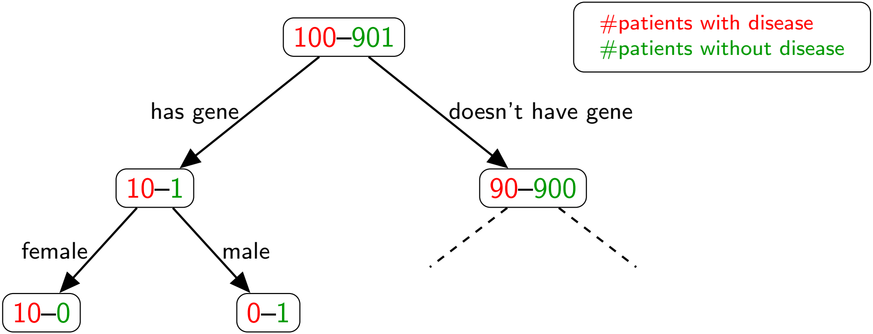
- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



$p(disease|has\text{-}gene \ \& \ male)$?

Why doing Hierarchical Smoothing?

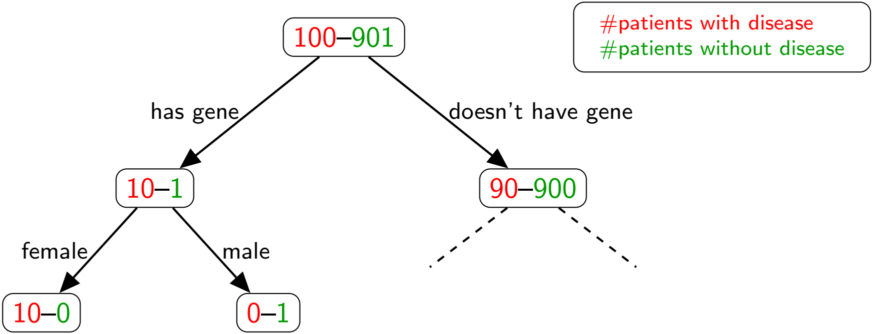
- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



$p_{MLE} = 0\%$

Why doing Hierarchical Smoothing?

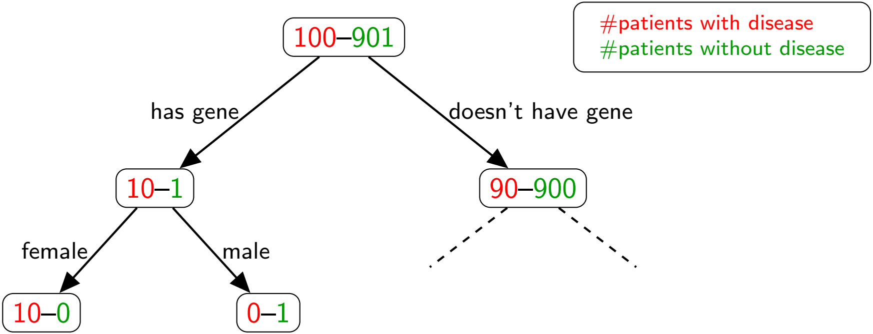
- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



$p_{Laplace} = 33\%$

Why doing Hierarchical Smoothing?

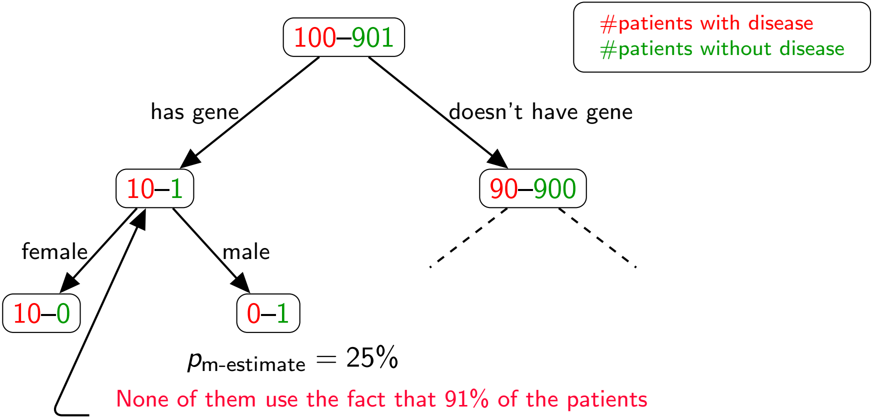
- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



$p_{\text{m-estimate}} = 25\%$

Why doing Hierarchical Smoothing?

- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females



$p_{\text{m-estimate}} = 25\%$

None of them use the fact that 91% of the patients with that gene have the disease!

Why doing Hierarchical Smoothing?

- You want to predict disease as a function of some rare gene G and sex, knowing that this disease is more prevalent for females

100–901

#patients with disease
#patients without disease

has gene

doesn't have gene

The idea of hierarchical smoothing/estimation is to make each node a function of the data at the node and the estimate at the parent.

$$p(\mathit{disease}|\text{has gene } \& \ \mathit{male}) \sim p(\mathit{disease}|\text{has gene})$$
$$p(\mathit{disease}|\text{has gene}) \sim p(\mathit{disease})$$

fema

10–0

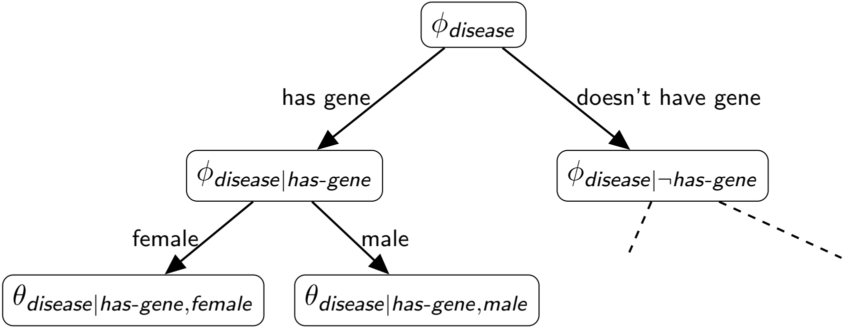
None of them use the fact that 91% of the patients with that gene have the disease!

Hierarchical Smoothing

Hierarchical Smoothing: When smoothing parameters in the context of a tree, use parent or ancestor parameters estimates in the smoothing.

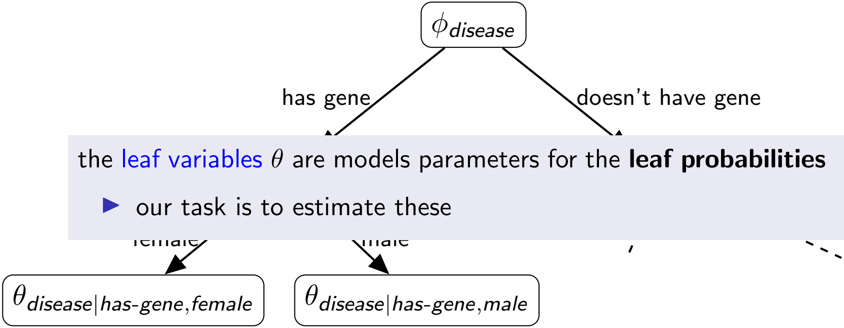
Hierarchical Smoothing

- You add prior **parameters** ϕ representing prior probability vectors for all ancestor nodes.



Hierarchical Smoothing

- You add prior **parameters** ϕ representing prior probability vectors for all ancestor nodes.

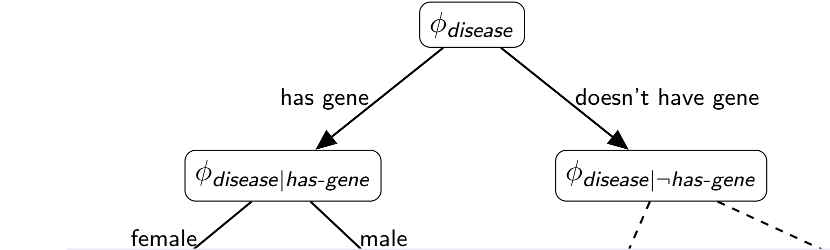


the leaf variables θ are models parameters for the **leaf probabilities**

- our task is to estimate these

Hierarchical Smoothing

- You add prior **parameters** ϕ representing prior probability vectors for all ancestor nodes.



the ancestor variables ϕ are **prior** parameters used in estimating the leaf probabilities

- these are **beliefs** not frequencies
- they do not correspond to frequencies at the ancestor nodes

Hierarchical Smoothing Model

Use **Dirichlet distributions** hierarchically.

- use $\mathrm{Dir}\left(\theta,\ \alpha\right)$ to represent a Dirichlet with parameter $\alpha\theta$
- normalised probability vector θ
- concentration (inverse variance) α

Hierarchical Smoothing Model

Use **Dirichlet distributions** hierarchically.

- use $\mathrm{Dir}\left(\theta,\ \alpha\right)$ to represent a Dirichlet with parameter $\alpha\theta$
- normalised probability vector θ
- concentration (inverse variance) α

Use the pattern:

$$\theta(\mathit{node}) \mid \phi(\mathit{node}) \quad \sim \quad \mathrm{Dir}\left(\phi(\mathit{parent}),\ \alpha(\mathit{node})\right)$$

Hierarchical Smoothing Model, cont.

Leaf probabilities:

$$\theta_{X_c|y,x_1,\cdots,x_n} \sim \mathrm{Dir}\left(\phi_{X_c|y,x_1,\cdots,x_{n-1}},\ \alpha_{y,x_1,\cdots,x_n}\right)$$

Hierarchical Smoothing Model, cont.

Leaf probabilities:

$$\theta_{X_c|y,x_1,\cdots,x_n} \sim \mathrm{Dir}\left(\phi_{X_c|y,x_1,\cdots,x_{n-1}},\ \alpha_{y,x_1,\cdots,x_n}\right)$$

Prior probabilities:

$$\begin{aligned}
\phi_{X_c} &\sim \mathrm{Dir}\left(\frac{1}{|X_c|}\vec{1},\ \alpha_0\right) \\
\phi_{X_c|y} &\sim \mathrm{Dir}\left(\phi_{X_c},\ \alpha_y\right) \\
&\cdots \\
\phi_{X_c|y,x_1,\cdots,x_{n-1}} &\sim \mathrm{Dir}\left(\phi_{X_c|y,x_1,\cdots,x_{n-2}},\ \alpha_{y,x_1,\cdots,x_{n-1}}\right)
\end{aligned}$$

Smoothing Formula

Smoothed probability estimates work back down the tree from the root using the pattern:

$$p(node) \quad \propto \quad count(node) + p(parent) \times \alpha(node)$$

Smoothing Formula

Smoothed probability estimates work back down the tree from the root using the pattern:

$$p(node) \quad \propto \quad count(node) + p(parent) \times \alpha(node)$$

Yielding:

$$\begin{aligned}
\hat{\phi}_{x_c} &= \frac{n_{x_c} + \frac{1}{|X_c|}\alpha_0}{n_{\cdot} + \alpha_0} \\
\hat{\phi}_{x_c|y,x_1,\cdots,x_i} &= \frac{n_{x_c|y,x_1,\cdots,x_i} + \hat{\phi}_{x_c|y,x_1,\cdots,x_{i-1}}\alpha_{y,x_1,\cdots,x_i}}{n_{\cdot|y,x_1,\cdots,x_i} + \alpha_{y,x_1,\cdots,x_i}} \\
\hat{\theta}_{x_c|y,x_1,\cdots,x_n} &= \frac{n_{x_c|y,x_1,\cdots,x_n} + \hat{\phi}_{x_c|y,x_1,\cdots,x_{n-1}}\alpha_{y,x_1,\cdots,x_n}}{n_{\cdot|y,x_1,\cdots,x_n} + \alpha_{y,x_1,\cdots,x_n}}
\end{aligned}$$

Smoothing Formula

Smoothed probability estimates work back down the tree from the root using the pattern:

$$p(node) \quad \propto \quad count(node) + p(parent) \times \alpha(node)$$

Yielding:

$$\begin{aligned}
\hat{\phi}_{x_c} &= \frac{n_{x_c} + \frac{1}{|X_c|}\alpha_0}{n. + \alpha_0} \\
\hat{\phi}_{x_c|y,x_1,\cdots,x_i} &= \frac{n_{x_c|y,x_1,\cdots,x_i} + \hat{\phi}_{x_c|y,x_1,\cdots,x_{i-1}}\alpha_{y,x_1,\cdots,x_i}}{n_{\cdot|y,x_1,\cdots,x_i} + \alpha_{y,x_1,\cdots,x_i}} \\
\hat{\theta}_{x_c|y,x_1,\cdots,x_n} &= \frac{n_{x_c|y,x_1,\cdots,x_n} + \hat{\phi}_{x_c|y,x_1,\cdots,x_{n-1}}\alpha_{y,x_1,\cdots,x_n}}{n_{\cdot|y,x_1,\cdots,x_n} + \alpha_{y,x_1,\cdots,x_n}}
\end{aligned}$$

But how do we get the estimates $\hat{\phi}_{x_c|y,x_1,\cdots,x_i}$?

Hierarchical Dirichlet

The Dirichlet distribution corresponds to a Dirichlet process with a discrete base distribution.

Hierarchical Dirichlet

The Dirichlet distribution corresponds to a Dirichlet process with a discrete base distribution.

We use a hierarchical Dirichlet processes (HDP) to handle the hierarchical Dirichlet distributions.

Historical Context for HDP

1990s-2003: Pitman and Ishwaran and James in mathematical statistics develop theory.

2006: Teh, Jordan, Beal and Blei develop HDP, *e.g.* applied to LDA.

2006-2011: Chinese restaurant processes (CRPs) go wild!

- require dynamic memory in implementation,

e.g. Chinese restaurant franchise, stick-breaking, *etc.*

But: **very slow, require large amounts of dynamic memory.**

Historical Context for HDP

1990s-2003: Pitman and Ishwaran and James in mathematical statistics develop theory.

2006: Teh, Jordan, Beal and Blei develop HDP, *e.g.* applied to LDA.

2006-2011: Chinese restaurant processes (CRPs) go wild!

- require dynamic memory in implementation,

e.g. Chinese restaurant franchise, stick-breaking, *etc.*

But: **very slow, require large amounts of dynamic memory.**

popularity of HDPs has decreased!

Historical Context for HDP, cont.

2011: Chen, Du, Buntine show slow methods not needed by introducing collapsed samplers.

2011: Buntine (unpublished) develops high performance algorithm for HDP and n-grams.

2014: Buntine and Mishra develop high performance algorithm for HDP and topic models.

Historical Context for HDP, cont.

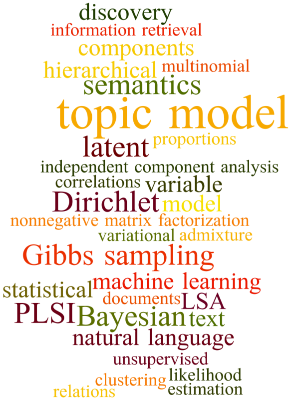
2011: Chen, Du, Buntine show slow methods not needed by introducing collapsed samplers.

2011: Buntine (unpublished) develops high performance algorithm for HDP and n-grams.

2014: Buntine and Mishra develop high performance algorithm for HDP and topic models.

- We use **high performance techniques** for the hierarchical Dirichlet process (HDP) to do inference.
  - outperforms Stochastic Variational Inference on some tasks
- This uses a (fairly) efficient Gibbs sampler.
  - no dynamic memory
  - with variable augmentation and caching
- Details in the paper.

Outline



Motivation

Bayesian Network Classifiers

Hierarchical Smoothing

Experimental Setup

Results

Conclusion

Main Claim

- Hierarchical smoothing
- applied to Bayesian network classifiers
- on **categorical datasets**
  - or pre-discretised attributes
- beats Random Forest

UCI Datasets

Domain	Case	Att	Class	Domain	Case	Att	Class
Connect-4Opening	67557	43	3	PimaIndiansDiabetes	768	9	2
Statlog(Shuttle)	58000	10	7	BreastCancer(Wisconsin)	699	10	2
Adult	48842	15	2	CreditScreening	690	16	2
LetterRecognition	20000	17	26	BalanceScale	625	5	3
MAGICGammaTelescope	19020	11	2	Syncon	600	61	6
Nursery	12960	9	5	Chess	551	40	2
Sign	12546	9	3	Cylinder	540	40	2
PenDigits	10992	17	10	Musk1	476	167	2
Thyroid	9169	30	20	HouseVotes84	435	17	2
Mushrooms	8124	23	2	HorseColic	368	22	2
Musk2	6598	167	2	Dermatology	366	35	6
Satellite	6435	37	6	Ionosphere	351	35	2
OpticalDigits	5620	49	10	LiverDisorders(Bupa)	345	7	2
PageBlocksClassification	5473	11	5	PrimaryTumor	339	18	22
Wall-following	5456	25	4	Haberman'sSurvival	306	4	2
Nettalk(Phoneme)	5438	8	52	HeartDisease(Cleveland)	303	14	2

and lots more datasets ... (not shown in the figure)

UCI Datasets Preprocessing

- Convert into ARFF format and process on WEKA.
- Apply the MDL discretization method of Fayyad and Irani.
- Also did one experiment with the very large Splice dataset.

Experimental Setup

- Use 5 runs of 2-fold cross validation.
  - known to be more stable than 10-fold cross validation

Experimental Setup

- Use 5 runs of 2-fold cross validation.
  - known to be more stable than 10-fold cross validation
- Evaluate with RMSE and 0-1 loss.
  - in classification context, MSE is related to the Brier score and is a proper scoring function, so evaluates the probabilities

Experimental Setup

- Use 5 runs of 2-fold cross validation.
  - known to be more stable than 10-fold cross validation
- Evaluate with RMSE and 0-1 loss.
  - in classification context, MSE is related to the Brier score and is a proper scoring function, so evaluates the probabilities
- Test the KDB versions and SKDB (max k=5) for HDP versus m-estimation with a back-off (for zero counts).
  - with m-estimation, we estimate m from $\{0, 0.05, 0.2, 1, 5, 20\}$ using cross validation on non-test subset

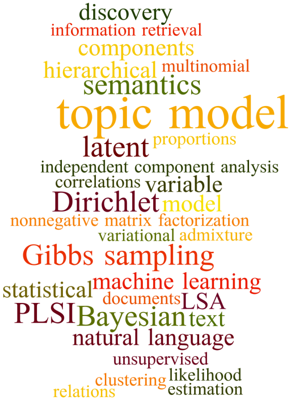
Experimental Setup

- Use 5 runs of 2-fold cross validation.
  - known to be more stable than 10-fold cross validation
- Evaluate with RMSE and 0-1 loss.
  - in classification context, MSE is related to the Brier score and is a proper scoring function, so evaluates the probabilities
- Test the KDB versions and SKDB (max k=5) for HDP versus m-estimation with a back-off (for zero counts).
  - with m-estimation, we estimate m from $\{0, 0.05, 0.2, 1, 5, 20\}$ using cross validation on non-test subset
- Also test against Random Forest with 100 trees.

Experimental Setup

- Use 5 runs of 2-fold cross validation.
  - known to be more stable than 10-fold cross validation
- Evaluate with RMSE and 0-1 loss.
  - in classification context, MSE is related to the Brier score and is a proper scoring function, so evaluates the probabilities
- Test the KDB versions and SKDB (max k=5) for HDP versus m-estimation with a back-off (for zero counts).
  - with m-estimation, we estimate m from $\{0, 0.05, 0.2, 1, 5, 20\}$ using cross validation on non-test subset
- Also test against Random Forest with 100 trees.
- Did one experiment with the very large Splice dataset.

Outline



- Motivation
- Bayesian Network Classifiers
- Hierarchical Smoothing
- Experimental Setup
- **Results**
- Conclusion

Main Claim

- Hierarchical smoothing
- applied to Bayesian network classifiers
- on categorical datasets
- **beats Random Forest**

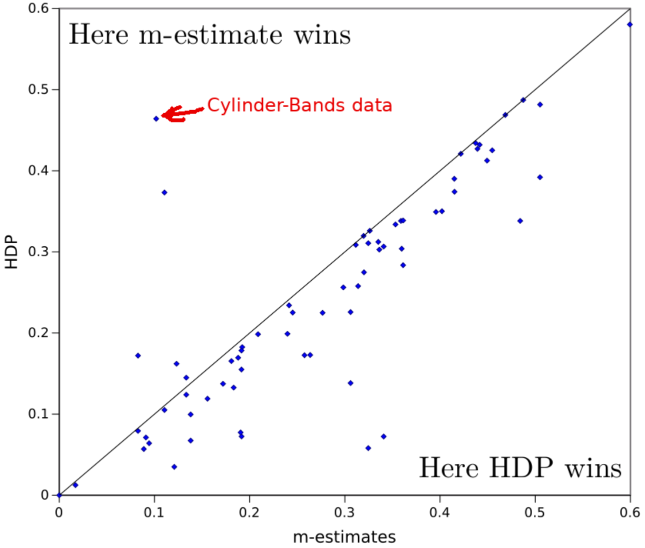
KDBs for HDP versus m-estimation

Classifier	Win–draw–loss for HDP vs m-estimate	
	0/1-loss	RMSE
Naive Bayes	**41–4–23**	40–0–28
TAN	**45–4–19**	**52–1–15**
kDB-1	**45–4–19**	**50–1–17**
kDB-2	**54–2–12**	**54–0–14**
kDB-3	**52–4–12**	**53–2–13**
kDB-4	**56–4– 8**	**56–0–12**
kDB-5	**60–4– 4**	**60–2– 6**
SkDB	**45–4–19**	**54–0–14**

HDP wins

* bold W-D-L values are significant at 5% by two-tailed binomial sign test

RMSE for KDB-5 for HDP versus m-estimation

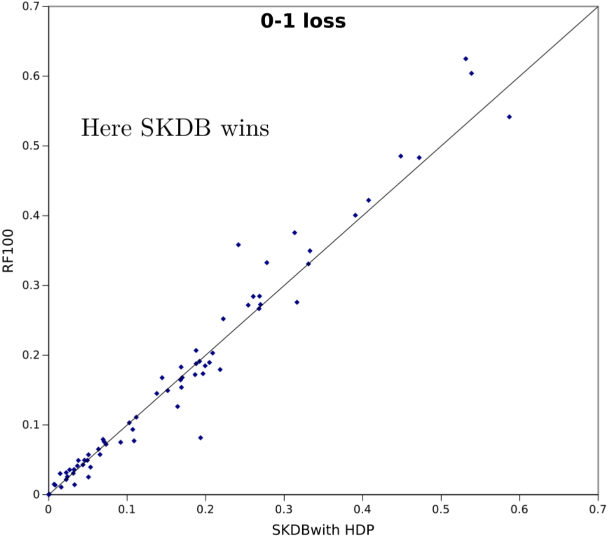


Comparison of TAN, SKDB and RF100

	Compared classifiers	Win–draw–loss	
		0/1-loss	RMSE
RF usually wins!	TAN-m vs RF	26–3–39	**25–0–43**
	SkDB-m vs RF	27–3–38	29–1–38
RF usually loses!	TAN-HDP vs RF	**42–3–23**	**42–0–26**
	SkDB-HDP vs RF	35–3–30	**44–0–24**

* bold W-D-L values are significant at 5% by two-tailed binomial sign test

0-1 Loss for SKDB-HDP versus RF100



SKDB versus Gradient Boosting

- Splice data: 50 million plus training data
- imbalanced: 1% positive class
- RF could not run with WEKA (out of memory)
- using XGBoost v0.6, 1 hour computation
- SKDB-HDP, 4 hour computation

Classifier	0/1-loss	RMSE
SkDB5-m	1.499%	0.1093
SkDB5-HDP	0.318%	0.0544
XGBoost	0.314%	0.0594

Outline

Motivation

Bayesian Network Classifiers

Hierarchical Smoothing

Experimental Setup

Results

Conclusion

Software for HDP Hierarchical Smoothing

Download, compile and run

```
git clone https://github.com/fpetitjean/HDP #download
cd HDP
ant #compile
java -jar jar/HDP.jar #run example
```

Example with your data

```
String [][]data = { // (stroke,weight,height)
  {"yes","heavy","tall"},
  ...
  {"yes","heavy","med"} };
ProbabilityTree hdp = new ProbabilityTree(); // init.
hdp.addDataset(data); //learn HDP tree - p(stroke|weight,height)
hdp.query("heavy","short"); //returns [61%, 39%]
hdp.query("heavy","tall"); //returns [31%, 69%]
hdp.query("light","tall"); //returns [9%, 91%]
```

Conclusion

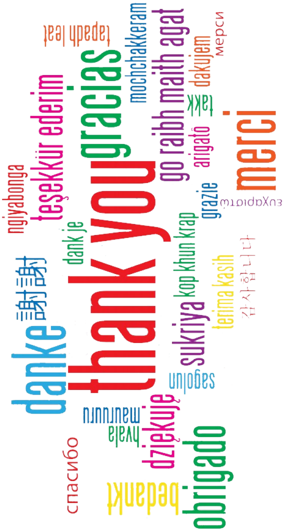
1. Hierarchical smoothing using HDP theory and algorithm presented.
   - HDP smoothing code on Github in Java

Conclusion

1. Hierarchical smoothing using HDP theory and algorithm presented.
   - HDP smoothing code on Github in Java
2. Combined HDP smoother with SKDB learner for BNCs to produce fast(-ish), scalable classification algorithm beating RFs.

Conclusion

1. Hierarchical smoothing using HDP theory and algorithm presented.
   - HDP smoothing code on Github in Java
2. Combined HDP smoother with SKDB learner for BNCs to produce fast(-ish), scalable classification algorithm beating RFs.
3. He ‘Penny’ Zhang (Monash PhD student) has significant improvements to the method.
   - sped up algorithm and beating Gradient Boosting of trees


thank you
gracias
danke
merci
obrigado
dziękuję
bedankt
спасибо
hvala
maururu
sagolun
sukriya
kop khun krap
terima kasih
grazie
arigatō
tack
dakujem
мерси
go raibh maith agat
mochchakkeram
tapadh leat
teşekkür ederim
ngiyabonga
dank je
謝謝
감사합니다
ευχαριστώ

Questions?