

Information theoretic model selection criteria in score-based Bayesian network structure learning

Tomi Silander
NAVER LABS Europe
Grenoble, France

May 28, 2018



Tomi Silander, NAVER LABS Europe and other stuff about BNSL May 28, 2018 1 / 72

Outline

- 1 Preliminaries
- 2 Motivation
- 3 Information theoretic criteria
- 4 Conclusion

Tomi Silander, NAVER LABS Europe and other stuff about BNSL May 28, 2018 3 / 72

NAVER LABS Europe (Grenoble)



- about 100 researchers (and growing) from 25 different countries
- machine learning and AI
 - graphical models, deep neural networks, optimization, reinforcement learning, ...
 - computer vision, natural language processing, robotics, mobility, ...

Tomi Silander, NAVER LABS Europe and other stuff about BNSL May 28, 2018 2 / 72

Outline

- 1 Preliminaries
- 2 Motivation
- 3 Information theoretic criteria
- 4 Conclusion

Tomi Silander, NAVER LABS Europe and other stuff about BNSL May 28, 2018 4 / 72

Super-exponentially many models

$$B_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{k=1}^n (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} B_{n-k} & \text{if } n > 0 \end{cases}$$

n	B_n^{-1}
1	1 000
2	0 333
3	0 040
4	0 00184
5	0 0000342
6	0 000000264
7	0 00000000878
8	0 000000000128
9	0 000000000000824
10	0 00000000000000240
11	0 0000000000000000316
12	0 000000000000000000192
13	0 00000000000000000000000535
14	0 000000000000000000000000000695
15	0 00000000000000000000000000000421
16	0 0000000000000000000000000000000119
17	0 000000000000000000000000000000000159
18	0 00000000000000000000000000000000000101
19	0 000301
20	0 00426

Model selection for what?

- Chickering and Heckerman, *A comparison of scientific and engineering criteria for Bayesian model selection*. Statistics and Computing, Vol10, Issue 1, pp 55–62 (January 2000)
- Scientific criterion** reveals the data generating mechanism: $P(S|D; \alpha)$
 - S = structure
 - D = data
 - α = hyperparameters
- Engineering criterion** finds the model that predicts well
 - $P(d|D, \hat{S}; \alpha)$
 - or even $P(d|D; \alpha) = \sum_S P(d|D, S; \alpha) P(S|D; \alpha)$.

Why select?

- Bayesians should not
 - neither for science nor for engineering
 - Especially since the most probable model has the probability 0.0000001.
 - so you “know” it’s wrong and select anyway?
- But what about “description” or compression
 - gonna send all the probabilities of all the models?
 - Interpretability:
 - but think Breiman’s *Statistical Modeling: The Two Cultures*
- Causality
 - cognitive arguments

Score-based vs. independence tests

It is often stated in papers tackling the task of inferring Bayesian network structures from data that there are these two distinct approaches:

- **Apply conditional independence tests when testing for the presence or otherwise of edges;**
- **Search the model space using a scoring metric.**

Here I argue that for complete data and a given node ordering this division is a myth, by showing that cross entropy methods for checking conditional independence are mathematically identical to methods based upon discriminating between models by their overall goodness-of-fit logarithmic scores.

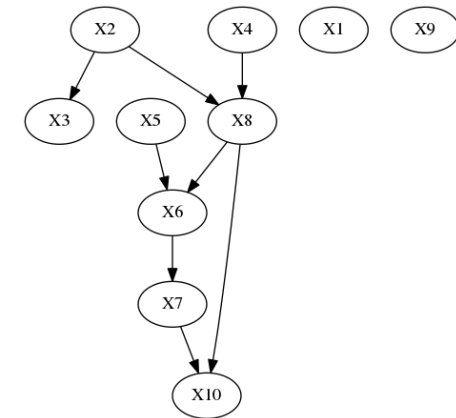
R. G. Cowell. *Conditions under which conditional independence and scoring methods lead to identical selection of Bayesian network models*. UAI 2001

Outline

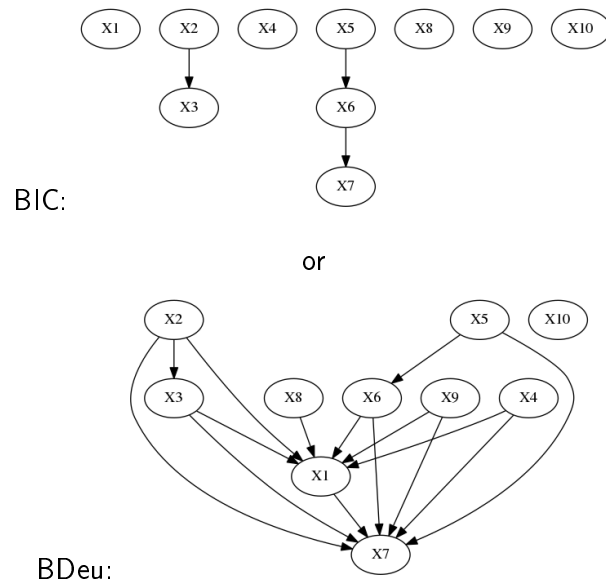
- 1 Preliminaries
- 2 Motivation
- 3 Information theoretic criteria
- 4 Conclusion

Here's my very practical problem

- Social scientists often have small sample sizes
 - Say ~60
- And when the ask me to learn a Bayesian network structure, I give them this:



Or this or this



Bayesian Dirichlet score

$$P(D|S; \alpha) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\sum_k \alpha_{ijk})}{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk} + n_{ijk})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})},$$

where

- n is number of categorical variables $X_1, \dots, X_n$
- r_i is the number of categorical values for variable X_i
- q_i is the number of value configurations of the parents Pa_i of i in the network S .
 - $q_i = \prod_{h \in Pa_i} r_h$, or $q_i = 1$, if $Pa_i = \emptyset$.
- n_{ijk} is the number of times the variable X_i has a value k when its parents have the j^{th} value configuration in a data D .
- $X_i | Pa_i = j \sim \text{Cat}(\theta_{ij})$, $\theta_{ij} \sim \text{Dir}(\alpha_{ij1}, \dots, \alpha_{ijr_i})$.

Parameter sensitivity of Bayesian Dirichlet score

$$P(D|S; \alpha) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\sum_k \alpha_{ijk})}{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk} + n_{ijk})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}.$$

- BDeu
 - $\alpha_{ijk} = \frac{\alpha}{q_i r_i}$
 - is known to be very sensitive to a single Dirichlet hyperparameter α .
 - And it is not “regular” (Suzuki)!!
- Some proposed alternatives with parameters give up likelihood equivalence (one of the main motivations for BDeu)
 - But be Bayesian $\alpha_{ijk} = \alpha$
 - or like Bayesian Dirichlet sparse BDs(α) a la $\alpha_{ijk} = \frac{\alpha}{\tilde{q}_i r_i}$, where $\tilde{q}_i$ is the number of parent configurations of Pa_i that actually occur in the data D at least once (Scutari).
 - empirically Bayes

Parameter sensitivity remedies

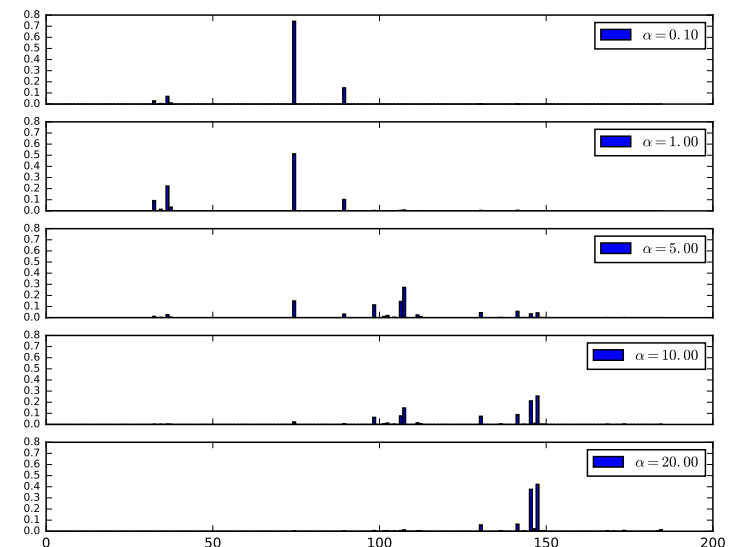
- Fixing α_{ijk} to some constant (like 1.0 or $\frac{1}{2}$) is one way to solve sensitivity.
- Using non-uniform structure prior can also help (?)
 - MU prior (marginal uniform): let it be *a priori* twice as probable to not have an arc than to have one (Scutari)
- Using criteria without parameters is an option
 - factorized Normalized Maximum Likelihood (fNML)
 - quotient Normalized Maximum Likelihood (qNML)
 - (discussed later)

But for now, let us look at the parameter sensitivity

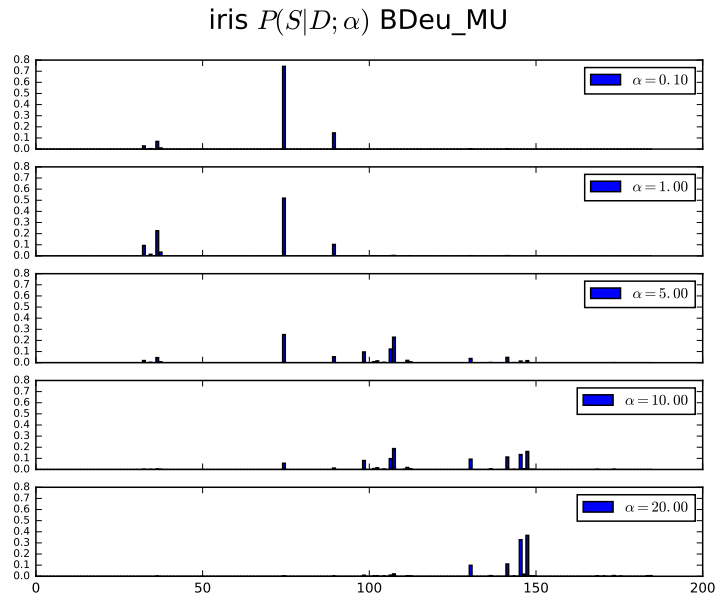
BDeu animation

BDeu posterior at $0.1 \leq \alpha \leq 20.00$

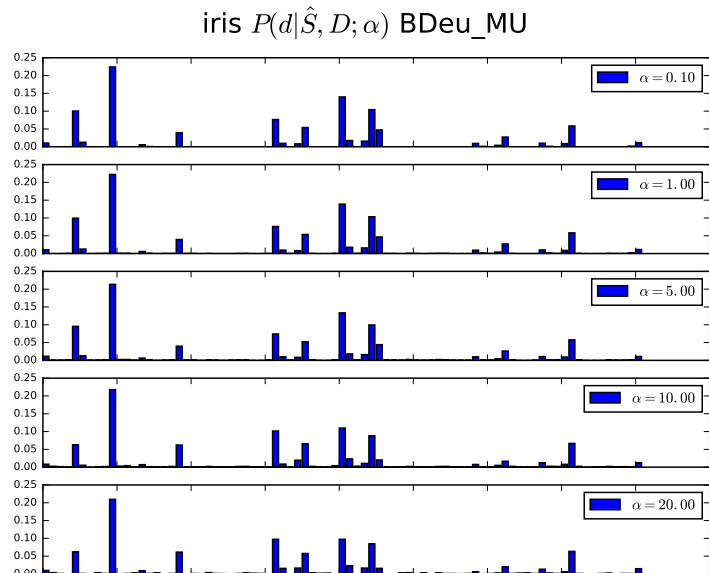
iris $P(S|D; \alpha)$ BDeu



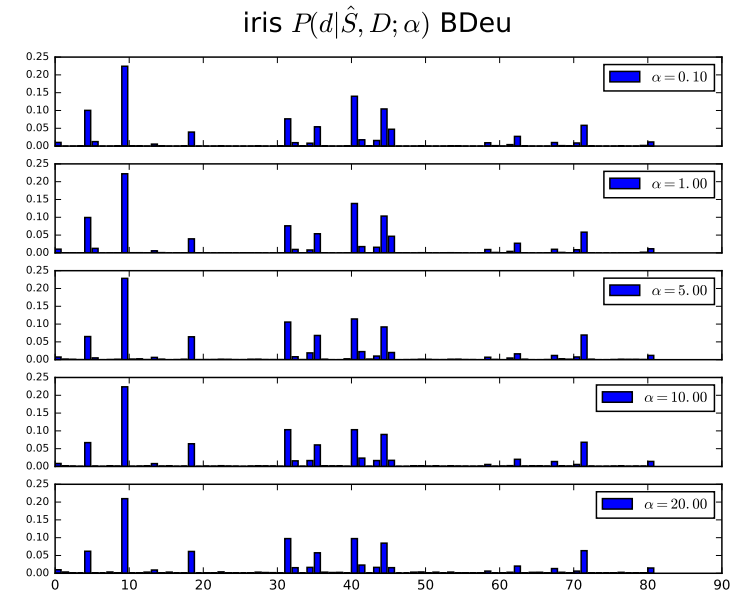
BDeu posterior $0.1 \leq \alpha \leq 20.00$ with MU



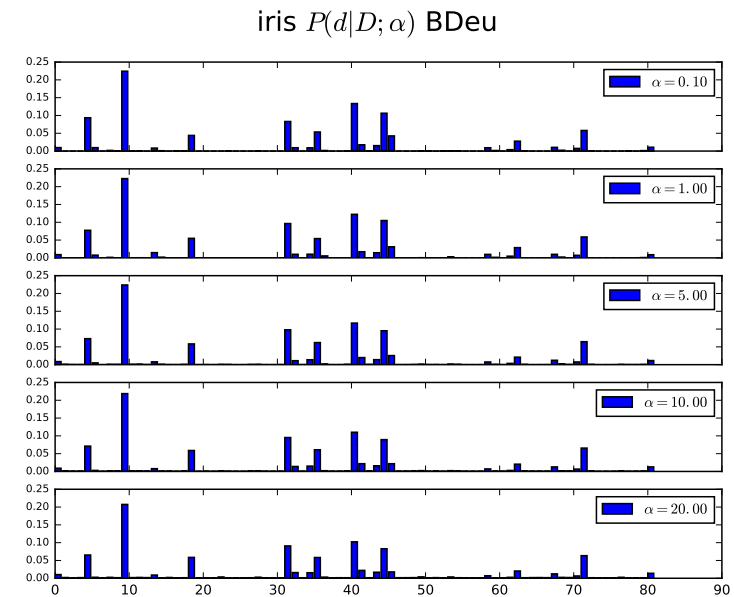
BDeu MAP predictive distributions $0.1 \leq \alpha \leq 20.00$ MU



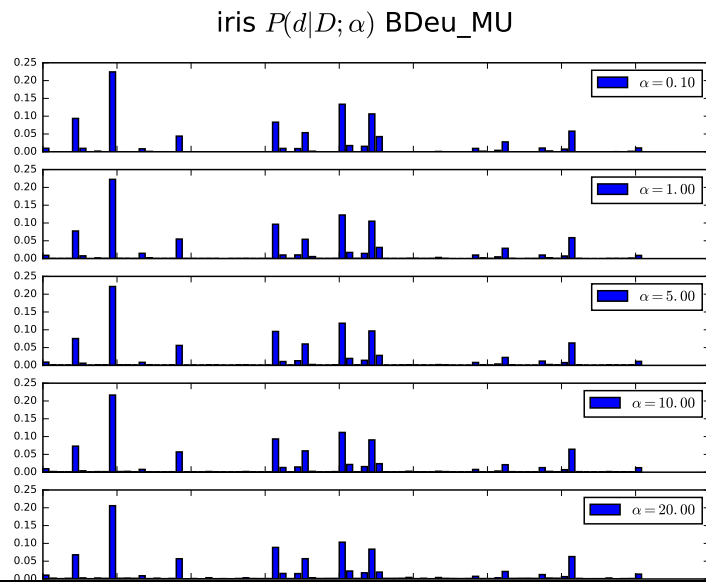
BDeu MAP predictive distributions $0.1 \leq \alpha \leq 20.00$



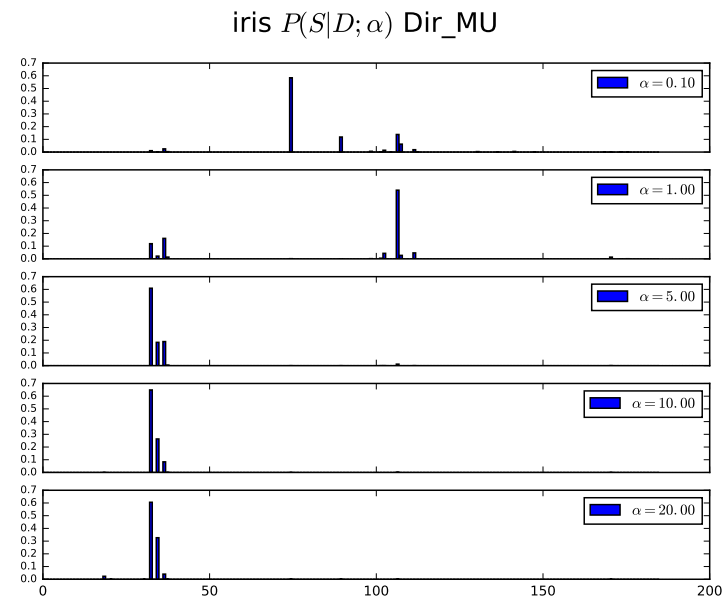
BDeu predictive distributions $0.1 \leq \alpha \leq 20.00$



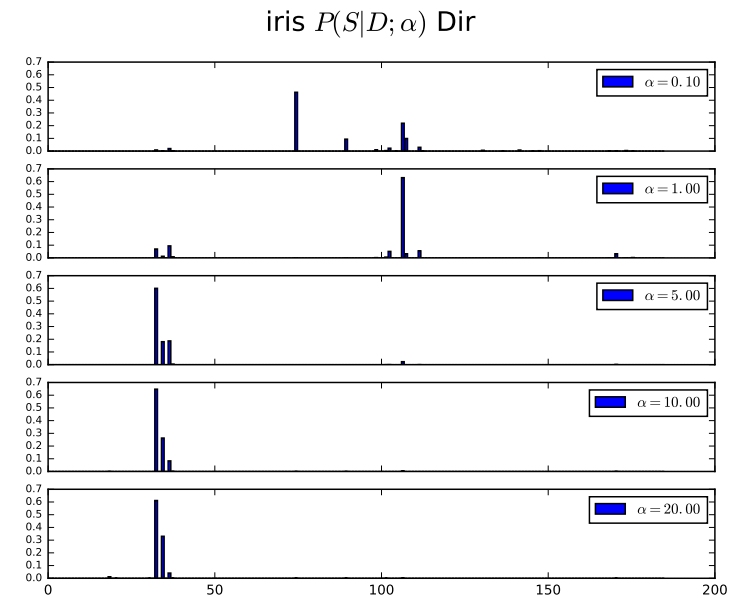
BDeu predictive distributions $0.1 \leq \alpha \leq 20.00$ with MU



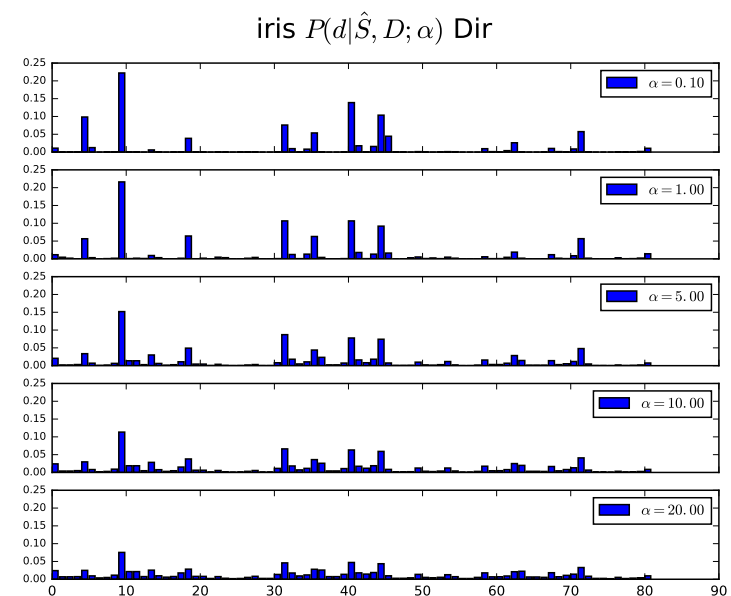
Dir posterior $0.1 \leq \alpha \leq 20.00$ with MU



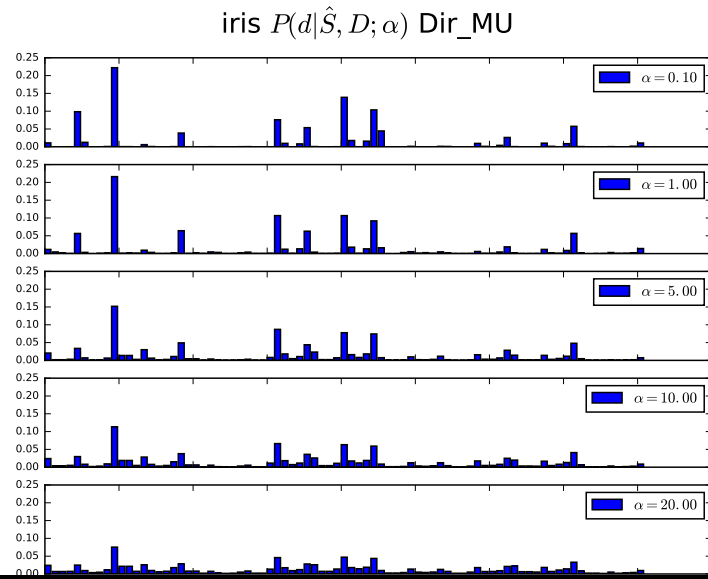
Dir posterior at $0.1 \leq \alpha \leq 20.00$



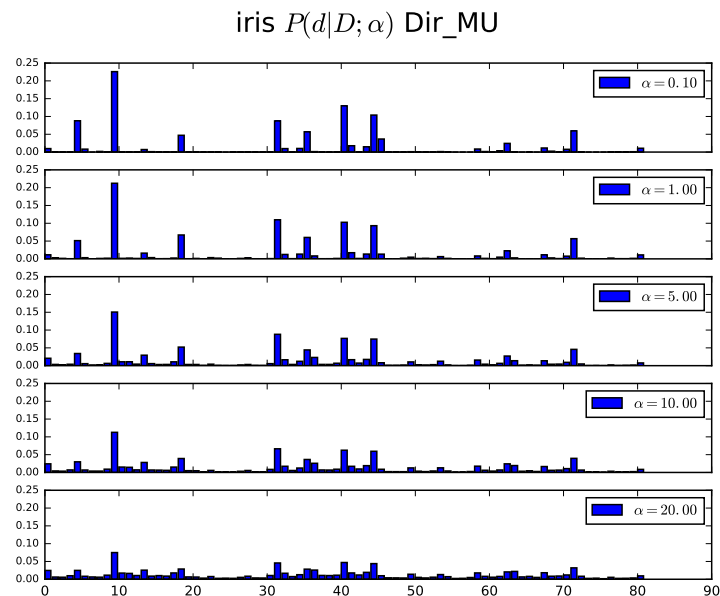
Dir MAP predictive distributions $0.1 \leq \alpha \leq 20.00$



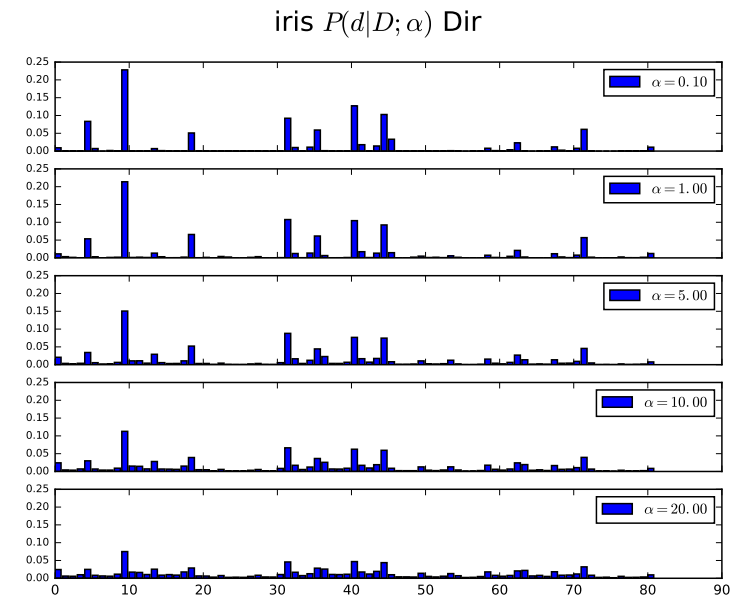
Dir MAP predictive distributions $0.1 \leq \alpha \leq 20.00$ MU



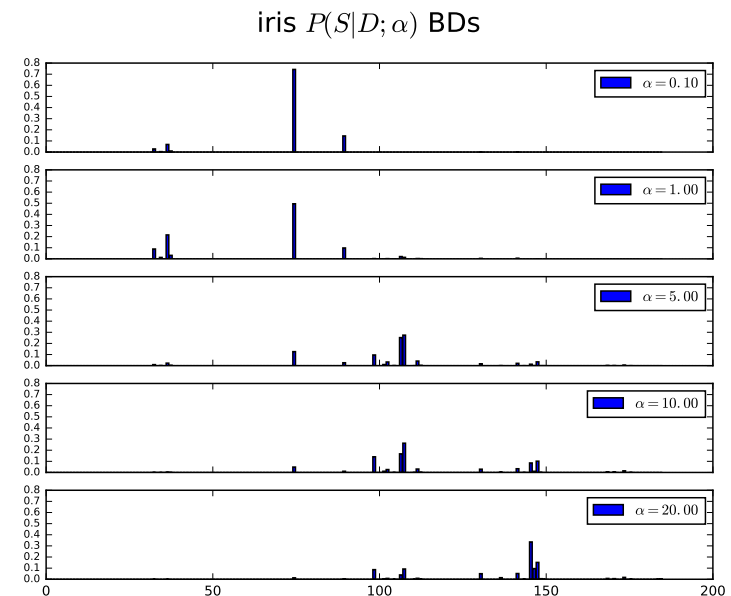
Dir predictive distributions $0.1 \leq \alpha \leq 20.00$ with MU



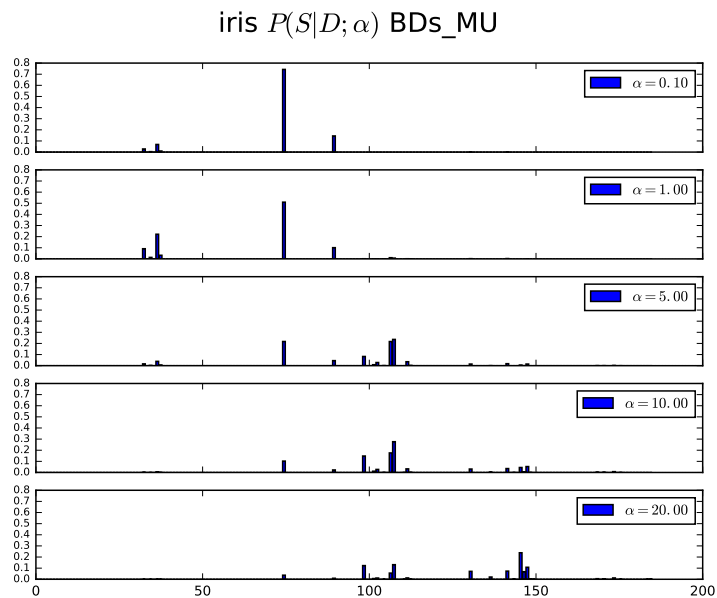
Dir predictive distributions $0.1 \leq \alpha \leq 20.00$



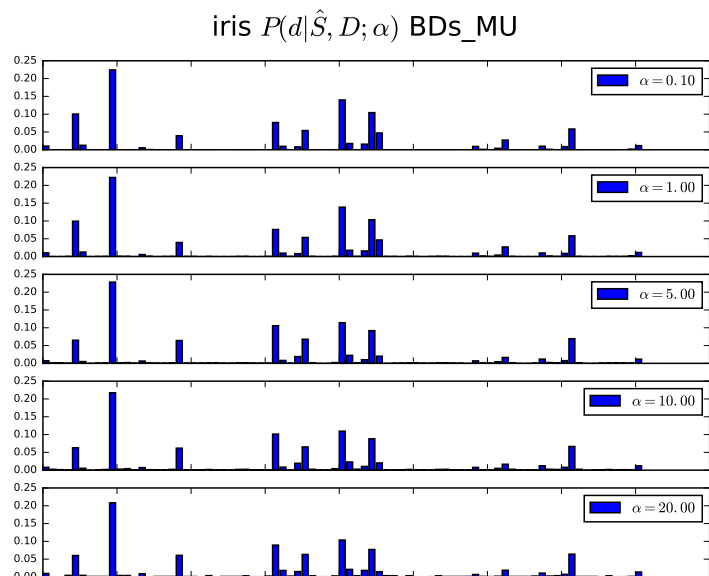
BDs posterior at $0.1 \leq \alpha \leq 20.00$



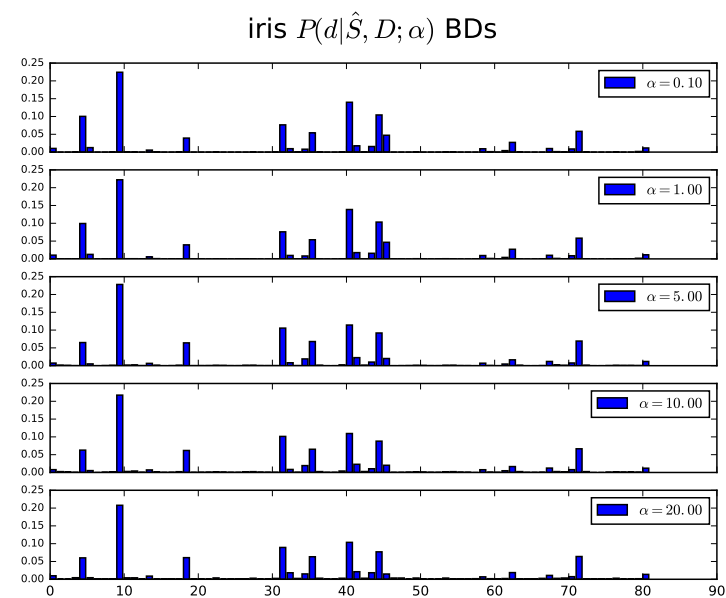
BDs posterior $0.1 \leq \alpha \leq 20.00$ with MU



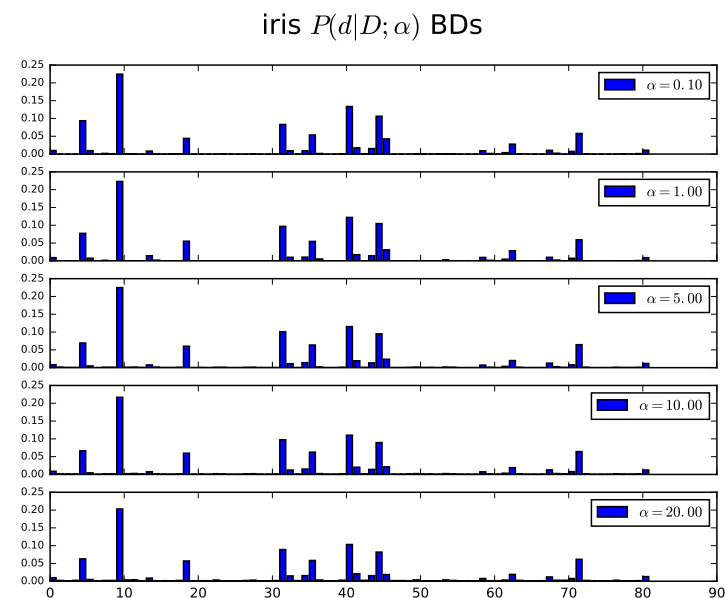
BDs MAP predictive distributions $0.1 \leq \alpha \leq 20.00$ MU



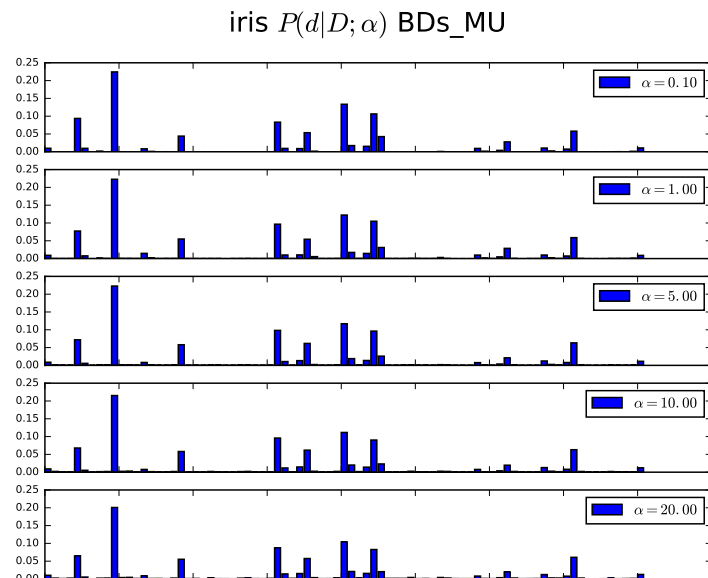
BDs MAP predictive distributions $0.1 \leq \alpha \leq 20.00$



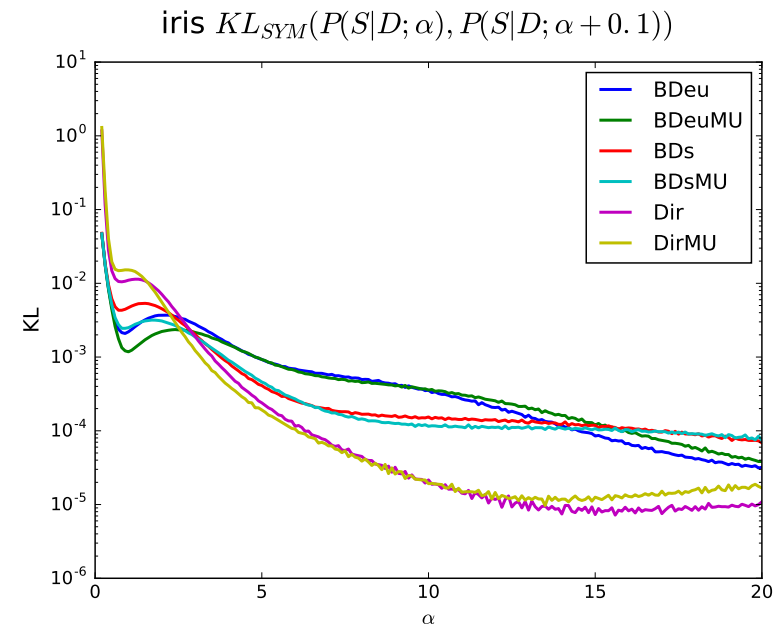
BDs predictive distributions $0.1 \leq \alpha \leq 20.00$



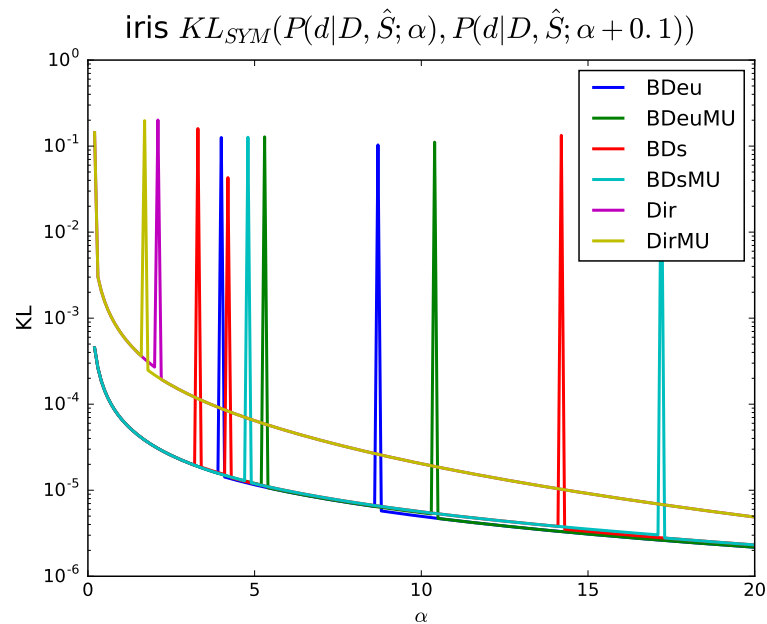
BDs predictive distributions $0.1 \leq \alpha \leq 20.00$ with MU



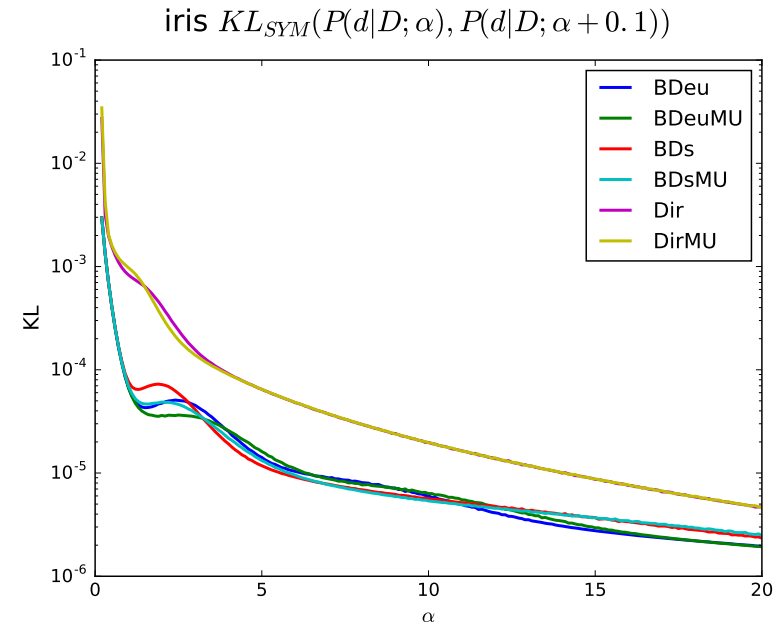
Comparing $P(S|D; \alpha)$ sensitivity



Comparing $P(d|D, \hat{S}; \alpha)$ sensitivity



Comparing $P(d|D; \alpha)$ sensitivity



Super exponentially many models

$$B_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{k=1}^n (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} B_{n-k} & \text{if } n > 0 \end{cases}$$

n	B_n^{-1}
1	1 000
2	0 333
3	0 040
4	0 00184
5	0 0000342
6	0 000000264
7	0 00000000878
8	0 000000000128
9	0 000000000000824
10	0 0000000000000240
11	0 00000000000000316
12	0 000000000000000192
13	0 0000000000000000535
14	0 00000000000000000695
15	0 000000000000000000421
16	0 0000000000000000000119
17	0 00000000000000000000159
18	0 000000000000000000000101
19	0 0000000000000000000000301
20	0 00000000000000000000000426

Sometimes with big data $P(\hat{S}|D) \approx 1$ (show that)

- Calculate the not normalized probability $cP(\hat{S}|D)$ and the not normalized probability $cP(S_2|D)$ of the second most probable network S_2 to get the ratio $r = \frac{P(S_2|D)}{P(\hat{S}|D)}$.
- Let B be the number of all the possible nets.
- Now

$$\begin{aligned} P(\hat{S}|D) &= \frac{P(\hat{S}|D)}{1} = \frac{P(\hat{S}|D)}{P(\hat{S}|D) + \sum_{S \neq \hat{S}} P(S|D)} \\ &\geq \frac{P(\hat{S}|D)}{P(\hat{S}|D) + (B-1)P(S_2|D)} \\ &= \frac{1}{1 + (B-1)r}. \end{aligned}$$

- sometimes r is small that $P(\hat{S}|D)$ is close to 1.0.
- adjust formula a bit for equivalent networks

Often $P(\hat{S}|D) \ll 1$ (how can you tell)

- Calculate the not normalized probabilities $cP(T_i)$ of likely structures $T = \{T_i\}$, other than MAP structure $\hat{S}$.
 - $cP(T) = \sum_i cP(T_i|D)$.
- Now even if all the other networks, i.e., those not in $T \cup \{\hat{S}\}$ had zero probability

$$P(\hat{S}|D) \leq \frac{cP(\hat{S}|D)}{cP(\hat{S}|D) + cP(T|D)}.$$

- often that alone makes $P(\hat{S}|D) \ll 1$
- people tend to get discouraged at $P(\hat{S}|D) \approx 1\%$.

Averaging over sub structures - not possible in general, but

- For small number on variables, one can learn the most probable order with dynamic programming and discrete Moebius transform.
 - Let $G \preceq G^*$ mean that G is a subnetwork of G^*

$$\begin{aligned} \text{setscore}(G^*) &= \sum_{G \subset G^*} \text{score}(G) \\ &= \sum_{G \subset G^*} \prod_{i=1}^n I_s(i, G_i) \\ &= \sum_{G_1 \subset G_1^*} \sum_{G_2 \subset G_2^*} \dots \sum_{G_n \subset G_n^*} \prod_{i=1}^n I_s(i, G_i) \\ &= \prod_{i=1}^n \left[\sum_{G_i \subset G_i^*} I_s(i, G_i) \right]. \end{aligned}$$

- The fast zeta transform is a way to compute a set function $F(V) = \sum_{W \subseteq V} f(W)$ for all sets $V \subseteq U$ in $O(n2^n)$ time.

when possible it works in practice, comparing

$$\frac{P(d|D, \alpha, 2^{G^*})}{P(d|D, \alpha, G^*)}$$

Data	BDeu 1.0	fNML
PostOperative	1.033 ± 0.052912	1.181 ± 0.046533
Iris	1.000 ± 0.000855	0.997 ± 0.004127
Wine	1.019 ± 0.014496	1.079 ± 0.022389
Glass	1.059 ± 0.030724	1.010 ± 0.009331
Thyroid	1.013 ± 0.008427	1.006 ± 0.003870
HeartStatlog	1.025 ± 0.013616	1.044 ± 0.016531
BreastCancer	0.997 ± 0.007784	1.039 ± 0.012359
HeartHungarian	1.008 ± 0.008601	1.047 ± 0.011981
HeartCleveland	1.006 ± 0.009991	1.042 ± 0.015379
Ecoli	0.998 ± 0.005293	1.008 ± 0.003127
Liver	1.004 ± 0.006049	1.017 ± 0.005205
Balance	1.000 ± 0.000000	1.000 ± 0.000000
BcWisconsin	1.008 ± 0.004476	1.020 ± 0.005898
Diabetes	1.002 ± 0.001896	1.008 ± 0.002771

Outline

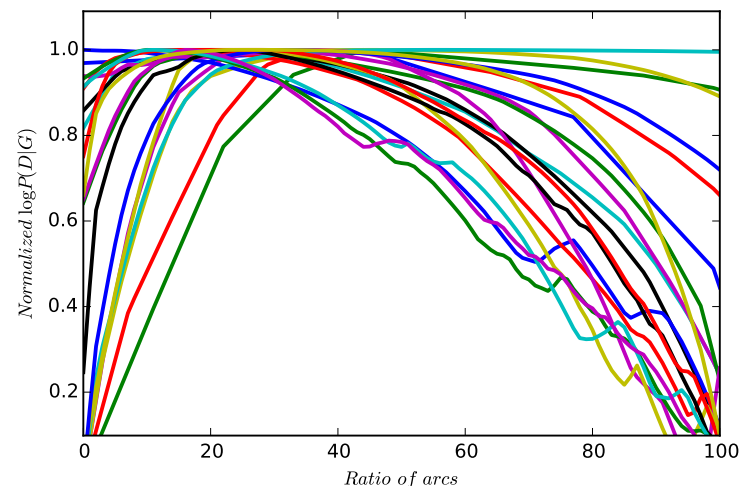
1 Preliminaries

2 Motivation

3 Information theoretic criteria

4 Conclusion

The posterior is flatter towards complex models



Thus learn a bit too complex model and average over substructures!

Parameter free alternatives based on code length

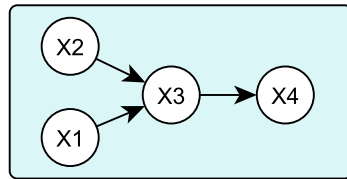
$$P_{NML}(D|S) = \frac{P(D|\hat{\theta}(D), S)}{\sum_{D'} P(D|\hat{\theta}(D), S)},$$

where D' goes over all the data sets that are of the same size as D .

- Impossible to naively compute for multivariate discrete data.
- But possible to compute for univariate categorical data.
- See, no hyperparameters.
- P_{NML} has the $\minmax_D$ regret property - never much worse than $post\ hoc\ P(D|\hat{\theta}(D), S)$
- fNML and qNML are attempts to use these ideas to derive practical approximations to P_{NML} .

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

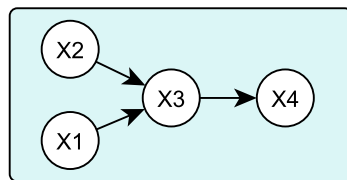


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_1 and X_2 being just blocks of binary and ternary data, this data can be coded (separately) by a Bayesian mixture or, better yet, by a normalized maximum likelihood code.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

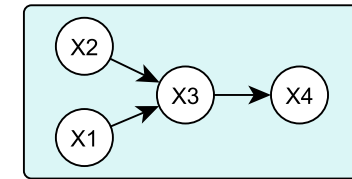


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

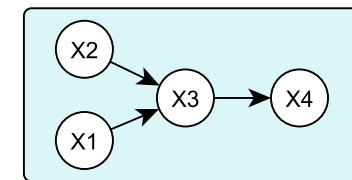


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_1 and X_2 being just blocks of binary and ternary data, this data can be coded (separately) by a Bayesian mixture or, better yet, by a normalized maximum likelihood code.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

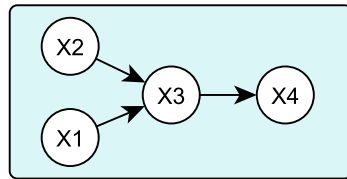


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

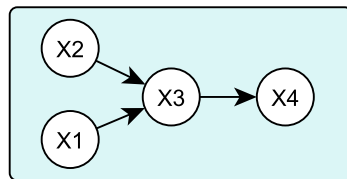


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

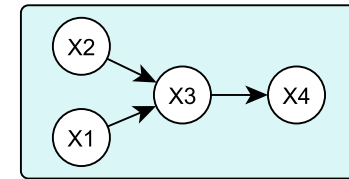


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

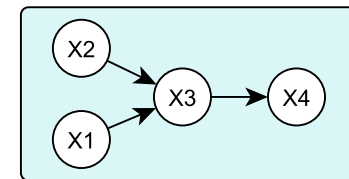


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

X_1	X_2	X_3	X_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	4
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			

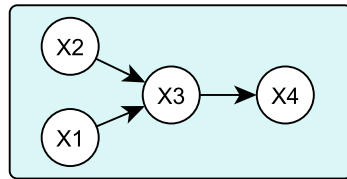


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_3 is then coded using $2 \times 3 = 6$ parts using 6 different codes, one per each possible value of (X_1, X_2) -pair.

Factorized codings

x_1	x_2	x_3	x_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			



$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

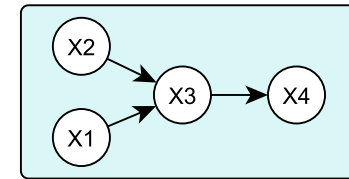
- X_4 is then coded in two parts, one per each possible value of $X_3 \in \{1, 2\}$.

from Bayes code to fNML code

- Coding the parts(!) using Bayes mixture is very sensitive to the the Dirichlet hyper parameter ($\alpha \in \mathbb{R}$)
 - $P(D|G, \alpha) = \int P(D|\theta)P(\theta|D)d\theta$
 - can be expressed in closed form
 - requirement that equivalent models get equivalent scores restricts the form of conjugate priors we can use
- coding the parts using NML appears to solve the problem
 - $P_{NML}^1(D; H) = \frac{P(D; \hat{\theta}(D), H)}{\sum_{D'} P(D'; \hat{\theta}(D'), H)}$
 - can be computed and approximated quickly
 - but the resulting criteria may give different scores for equivalent models

Factorized codings

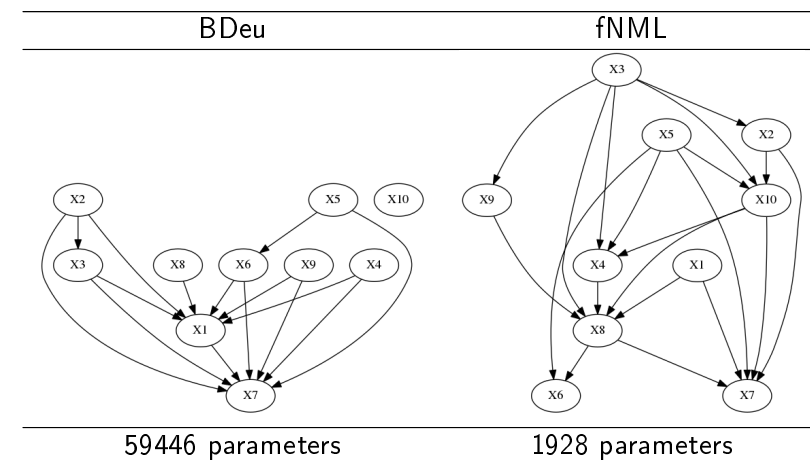
x_1	x_2	x_3	x_4
1	2	2	1
2	3	2	4
2	3	2	4
1	1	1	3
2	1	1	2
1	2	2	2
2	3	2	2
1	2	1	1
1	3	2	1
1	1	1	1
2	1	2	3
2	3	2	3
2	3	1	3
2	2	1	2
2	2	1	2
1	3	1	2
1	2	2	1
1	2	2	1
2	1	2	1
2	2	1	2
1	2	1	4
1	3	1	4
2	3	1	3
2	2	2	1
r= 2 3 2 4			



$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- X_4 is then coded in two parts, one per each possible value of $X_3 \in \{1, 2\}$.

Example models



- -Breast cancer data with $N = 70$, $n = 10$

quotient normalized maximum likelihood (qNML)

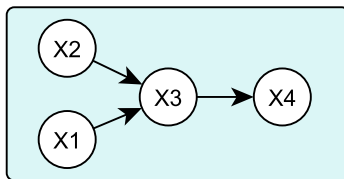
- Computing NML for the whole data D is not feasible because of the normalizer
 - for our 24×4 example matrix we have to sum over $2^{24} 3^{24} 2^{24} 4^{24}$ different data matrices
- in qNML, like in fNML, we use the fact that there are ways to compute NML for a single categorical data vector efficiently

$$S_{qNML}(D; H) := \prod_{i=1}^n \frac{P_{NML}^1(D_{\cdot i}, D_{\cdot \pi_i}; H)}{P_{NML}^1(D_{\cdot \pi_i}; H)}$$

- to code several data columns with P_{NML}^1 , we first collapse their value combinations into a single categorical value!

qNML - column X_2

	X_1	X_2	X_3	X_4
$\bar{x}$	1	2	2	1
$\bar{y}$	2	3	2	4
$\bar{z}$	2	3	2	4
$\bar{w}$	1	1	1	3
$\bar{v}$	2	1	1	2
$\bar{u}$	1	2	2	2
$\bar{t}$	2	3	2	2
$\bar{s}$	1	2	1	1
$\bar{r}$	1	3	2	1
$\bar{q}$	1	1	1	1
$\bar{p}$	2	1	2	3
$\bar{o}$	2	3	2	3
$\bar{n}$	2	3	1	3
$\bar{m}$	2	2	1	2
$\bar{l}$	2	2	1	2
$\bar{k}$	1	3	1	2
$\bar{j}$	1	2	2	1
$\bar{i}$	1	2	2	1
$\bar{h}$	2	1	2	1
$\bar{g}$	2	2	1	2
$\bar{f}$	1	2	1	4
$\bar{e}$	1	3	1	4
$\bar{d}$	2	3	1	3
$\bar{c}$	2	2	2	1
$\bar{b}$	2	3	2	4

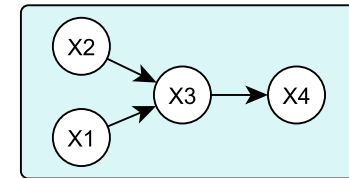


$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- With no parents, the columns X_1 and X_2 are coded (separately) by a normalized maximum likelihood code.

qNML - column X_1

	X_1	X_2	X_3	X_4
1	2	2	2	1
2	3	2	2	4
3	2	3	2	4
1	1	1	1	1
2	1	1	2	2
1	2	2	2	2
2	3	2	2	2
1	2	1	1	1
1	3	2	1	1
1	1	1	1	1
2	1	2	3	3
2	3	2	3	3
2	2	1	2	2
2	2	1	2	2
1	3	1	2	2
1	2	2	1	1
1	2	2	1	1
2	1	2	1	1
2	1	2	1	4
1	3	1	1	4
2	3	1	3	3
2	2	2	2	1
2	3	2	2	4



$$P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_3)$$

- With no parents, the columns X_1 and X_2 are coded (separately) by a normalized maximum likelihood code.

qNML - column X_3

- Collapsing (X_1, X_2, X_3) , and (X_1, X_2) to code them by P_{NML}^1 .

	X_1	X_2	X_3	X_4
1	2	2		
2	3	2		
2	3	2		
1	1	1		
2	1	1		
1	2	2		
2	3	2		
1	2	1		
1	3	2		
1	1	1		
2	1	2		
2	3	2		
2	3	1		
2	2	1		
2	2	1		
1	3	1		
1	2	2		
1	2	2		
2	1	2		
2	2	1		
1	2	1		
1	3	1		
2	3	1		
2	2	2		
$r=$	2	3	2	4

	X_1	X_2	X_3
	1	2	2
	2	3	2
	2	3	2
	1	1	1
	2	1	1
	1	2	2
	2	3	2
	1	2	1
	1	3	2
	1	1	1
	2	1	2
	2	3	2
	2	3	1
	2	2	1
	2	2	1
	1	3	1
	1	2	2
	1	2	2
	2	2	1
	1	2	1
	1	3	1
	2	3	1
	2	2	2
$r =$	2	3	2

	X_1	X_2	X_3	X_4
1	2		2	
2	3		2	
2	3		2	
1	1		1	
2	1		1	
1	2		2	
2	3		2	
2	2		1	
1	3		2	
1	1		1	
2	1		2	
2	3		2	
2	3		1	
2	2		1	
1	3		1	
1	2		2	
1	2		2	
2	2		1	
1	2		1	
2	3		1	
2	2		1	
$r=$	2	3	2	

	X_1	X_2	X_3	X
	4			
	12			
	12			
	1			
	7			
	4			
	12			
	3			
	6			
	1			
	8			
	12			
	11			
	9			
	5			
	4			
	4			
	8			
	9			
	3			
	5			
	11			
	10			
$r =$	12	4		

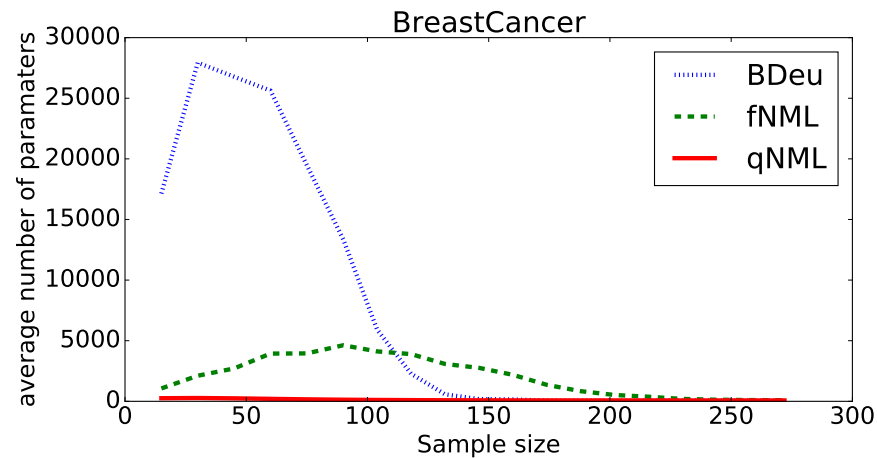
	X_1	X_2	X_3	X_4
2	2	2	1	
6		2	4	
6		2	4	
1		1	3	
4		1	2	
2		2	2	
6		2	2	
2		1	1	
3		2	1	
1		1	1	
4		2	3	
6		2	3	
6		1	3	
5		1	2	
5		1	2	
3		1	2	
2		2	1	
2		2	1	
4		2	1	
5		1	2	
2		1	4	
3		1	4	
6		1	3	
5		2	1	
Σ	6	2	4	

qNML - column X_4

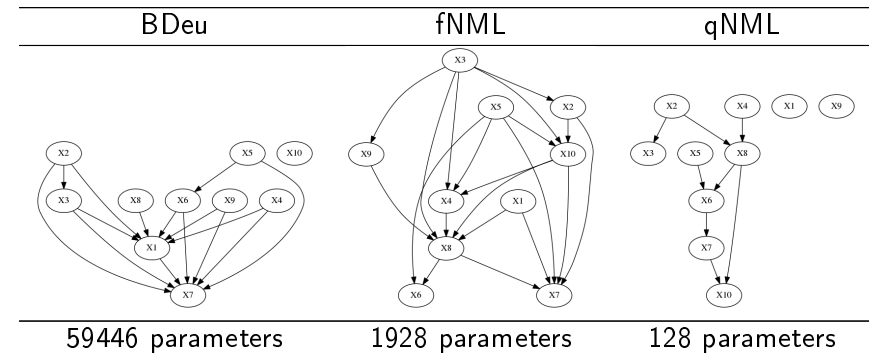
- Collapsing X_3 and X_4 to code them by P_{NML}^1 .

X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r	X_1	X_2	X_3	X_4	r
1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1	
2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4	
2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4	
1	1	1	3		1	1	1	3		1	1	1	3		1	1	1	3		1	1	1	3		1	1	1	3		1	1	1	3	
2	1	1	2		2	1	1	2		2	1	1	2		2	1	1	2		2	1	1	2		2	1	1	2		2	1	1	2	
1	2	2	2		1	2	2	2		1	2	2	2		1	2	2	2		1	2	2	2		1	2	2	2		1	2	2	2	
2	3	2	2		2	3	2	2		2	3	2	2		2	3	2	2		2	3	2	2		2	3	2	2		2	3	2	2	
1	2	1	1		1	2	1	1		1	2	1	1		1	2	1	1		1	2	1	1		1	2	1	1		1	2	1	1	
1	3	2	1		1	3	2	1		1	3	2	1		1	3	2	1		1	3	2	1		1	3	2	1		1	3	2	1	
1	1	1	1		1	1	1	1		1	1	1	1		1	1	1	1		1	1	1	1		1	1	1	1		1	1	1	1	
2	1	2	3		2	1	2	3		2	1	2	3		2	1	2	3		2	1	2	3		2	1	2	3		2	1	2	3	
2	3	2	3		2	3	2	3		2	3	2	3		2	3	2	3		2	3	2	3		2	3	2	3		2	3	2	3	
2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3	
2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2	
2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2	
1	3	1	2		1	3	1	2		1	3	1	2		1	3	1	2		1	3	1	2		1	3	1	2		1	3	1	2	
1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1	
1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1	
1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1		1	2	2	1	
2	1	2	1		2	1	2	1		2	1	2	1		2	1	2	1		2	1	2	1		2	1	2	1		2	1	2	1	
2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2		2	2	1	2	
1	2	1	4		1	2	1	4		1	2	1	4		1	2	1	4		1	2	1	4		1	2	1	4		1	2	1	4	
1	3	1	4		1	3	1	4		1	3	1	4		1	3	1	4		1	3	1	4		1	3	1	4		1	3	1	4	
2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3		2	3	1	3	
2	2	2	1		2	2	2	1		2	2	2	1		2	2	2	1		2	2	2	1		2	2	2	1		2	2	2	1	
2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4		2	3	2	4	

Number of parameters by sample size

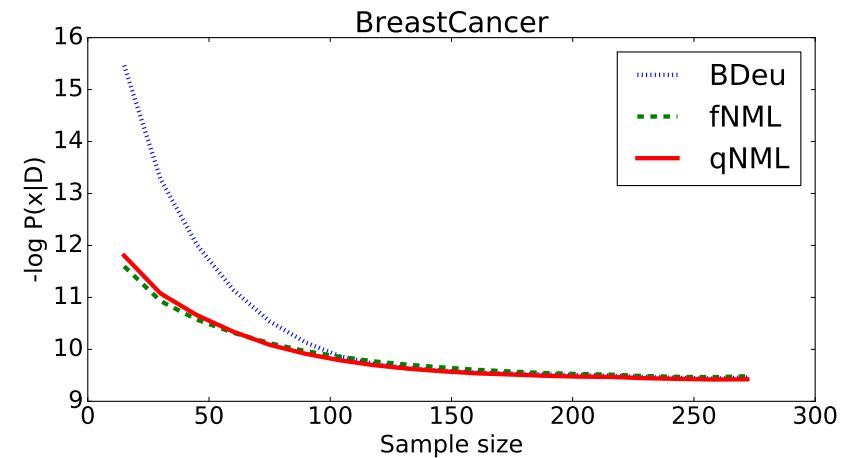


Example models



- : Breast cancer data with $N = 70$, $n = 10$

Predictive log-loss by sample size



Which parameters for fNML and qNML (and BIC/MDL)?

- It would be odd to suggest hyperparameters for parameter estimation if we argue for getting rid of them in structure learning
- Sequential normalized maximum likelihood
 - one step lookahead NML

$$P(d|D, S) = \frac{P(d, D|\hat{\theta}(d, D), S)}{\sum_{d'} P(d', D|\hat{\theta}(d', D), S)}.$$

- For univariate categorical variable this leads to using the parameters:

$$\theta_{ijk} \propto e(n_{ijk})(n_{ijk} + 1),$$

where $e(n) = \left(\frac{n+1}{n}\right)^n$.

- kind of weighted Laplace smoothing

qNML pedictive results (bit bigger data)

Table: Average predictive performance rank over different sample sizes for different model selection criteria in 20 different data sets.

Data	N	BDeu BPP	BIC sNML	fNML sNML	qNML sNML
Balance	625	<u>3.35</u>	1.91	1.59	3.16
BcWisco	699	<u>3.06</u>	2.03	2.89	2.02
Diabete	768	<u>2.91</u>	2.70	2.68	1.71
TicTacT	958	<u>3.44</u>	2.71	1.31	2.53
Yeast	1484	2.60	<u>3.76</u>	1.55	2.10
Abalone	4177	2.60	<u>3.64</u>	1.04	2.72
PageBlo	5473	2.24	<u>3.61</u>	1.31	2.83
Adult	32561	3.23	<u>3.77</u>	1.00	2.00
Shuttle	58000	1.44	<u>3.78</u>	1.56	3.22

qNML pedictive results (small data)

Table: Average predictive performance rank over different sample sizes for different model selection criteria in 20 different data sets.

Data	N	BDeu BPP	BIC sNML	fNML sNML	qNML sNML
PostOpe	90	2.79	1.20	<u>3.06</u>	2.94
Iris	150	<u>2.82</u>	2.37	2.27	2.54
Wine	178	<u>3.23</u>	1.88	2.67	2.22
Glass	214	<u>3.61</u>	3.09	1.42	1.88
Thyroid	215	2.55	<u>3.21</u>	1.80	2.44
HeartSt	270	<u>3.12</u>	1.39	3.12	2.37
BreastC	286	<u>3.09</u>	1.41	2.97	2.53
HeartHu	294	<u>3.18</u>	1.66	2.90	2.27
HeartCl	303	<u>3.46</u>	1.38	2.99	2.17
Ecoli	336	3.20	<u>3.53</u>	1.24	2.04
Liver	345	<u>3.17</u>	2.39	2.69	1.75

qNML properties

- qNML does not have tunable hyper parameters
- qNML yields more parsimonious models than BDeu and fNML
- qNML is still competitive predictively
- qNML is as quick/slow to compute than BDeu and fNML
- qNML gives equal scores to equal dependency hypotheses
- qNML is consistent and regular
- qNML coincides with NML for exponentially many models

Outline

- 1 Preliminaries
- 2 Motivation
- 3 Information theoretic criteria
- 4 Conclusion

Summary

- Out of “common” scoring criteria BIC and qNML have all the “desirable properties”
 - BIC is based on asymptotic approximation of marginal likelihood
 - or two part code (sometimes stacking called MDL)

	BIC	BDeu	BDs	fNML	qNML	NML
decomposable	x	x	x	x	x	
consistent	x	x	x	x	x	x
LH equivalent	x	x			x	x
Regular	x		x	x	x	x
Hyperparameter free	x			x	x	x

Table: Features of scoring criteria

Summary

- The parameter sensitivity is not just about the selected network
 - but the whole posterior distribution
- The parameter sensitivity is still a problem
 - blessing
- I still do not know what model to give to a practitioner
 - but I tend to check the parameter free options first
 - and also check for the probability of other models to have an idea how probable my model is

Thanks

Thank You!