Learning Hybrid Bayesian Networks by MML

Rodney T. O'Donnell, Lloyd Allison, and Kevin B. Korb

School of Information Technology
Monash University
Clayton, Victoria 3800, Australia

Abstract. We use a Markov Chain Monte Carlo (MCMC) MML algorithm to learn hybrid Bayesian networks from observational data. Hybrid networks represent local structure, using conditional probability tables (CPT), logit models, decision trees or hybrid models, i.e., combinations of the three. We compare this method with alternative local structure learning algorithms using the MDL and BDe metrics. Results are presented for both real and artificial data sets. Hybrid models compare favourably to other local structure learners, allowing simple representations given limited data combined with richer representations given massive data.

1 Introduction

There is a large literature on methods of learning Bayesian networks from observed data. Much of that work has focused solely on learning network structure, treating network parameterization as a separate process. However, some work has been done on learning network structure and parameters simultaneously and many algorithms exist for performing this task. Most techniques involve a heuristic search through network space to find the optimal combination of directed acyclic graph (DAG) and the set of associated conditional probability tables (CPTs).

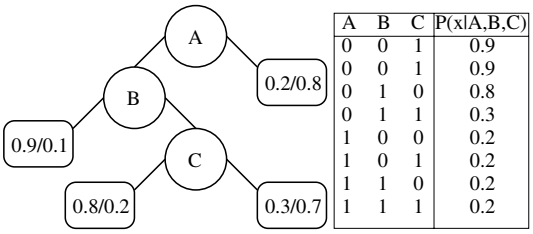
For discrete networks, CPTs are the most powerful representation of a child node's probability distribution. Any variety of interaction between parent states may be expressed (where the parameters are entirely independent of each other); likewise, any variety of functional dependence between parameters may be expressed, such as noisy-OR models. When, as in this last case, some parameters are highly dependent upon others, this is described as *local structure*. The expressive power, and complexity, of CPTs is wasted in such cases, and so there is value in finding simpler representations, such as modest-sized decision trees.

For example, consider Figure 1, which shows a CPT requiring 8 continuous parameters; expressed in a decision tree form it requires only four. The benefit of non-CPT models quickly becomes apparent as more parent variables are added. The advantage of using local structures that are more economical than CPTs in Bayesian networks has been clearly shown in [1,2] and elsewhere.

Here we apply CaMML (Causal discovery via MML) [3,4,5] to the learning of local structure in Bayesian networks in an especially flexible way, using either full CPTs, logit models or decision trees, or any combination of these determined

A. Sattar and B.H. Kang (Eds.): AI 2006, LNAI 4304, pp. 192–203, 2006.

on a node-by-node basis (hybrid models). We compare our approach with Nir Friedman's implementation[1] of the Bayesian BDe metric[6] and the MDL[7] metric, which were also applied to learning local structure using decision trees, although without hybrid model learning.



A	B	C	P(x|A,B,C)
0	0	1	0.9
0	0	1	0.9
0	1	0	0.8
0	1	1	0.3
1	0	0	0.2
1	0	1	0.2
1	1	0	0.2
1	1	1	0.2

Fig. 1. Decision Tree and CPT example

Previous work in learning local structure with MML used logit models[2]. Here we extend CaMML to decision trees, making it possible to compare the results effectively with local structure learning elsewhere. This requires a new coding technique, distinct from that of [8], since the decision trees involved in local structure have unique constraints. The hybrid learning likewise extends earlier work, allowing all varieties of local structure to be represented in forms suitable to their complexity. This provides an effective means of automatically adapting representational complexity to the amount of data available.

2 Metrics

Minimum Message Length (MML) inference is a method of estimating a fully parameterized model, using Bayes's Theorem: [9]

$$P(H\&D) = P(H).P(D|H) = P(D).P(H|D)$$

for a hypothesis (e.g., a Bayesian network), H, and data, D, and on Shannon's law for optimal codes [10]

$$msgLen(E) = -\log P(E)$$

requiring an event, E, to have a code of length $-\log P(E)$. Lengths can be measured in bits (log base 2) or, if mathematically convenient, in nits (natural logs). In any case,

$$msgLen(H\&D) = msgLen(H) + msgLen(D|H)$$

Being Bayesian, MML requires an explicit prior distribution, $P(H)$, on hypotheses. If the hypothesis space is discrete, a hypothesis has a non-zero probability and computing a message length is in principle straightforward, involving a negative log probability and a negative log likelihood. If however a hypothesis has one or more continuous parameters, it has no probability as such, rather a probability density. MML requires that continuous parameters be stated to optimal, *finite* precision. The latter point is subtle, allowing Bayes's rule to be

applied to continuous hypothesis spaces. MML inference is consistent and is invariant under monotonic transformations of parameters.

MML uses an enumerable code-book of hypotheses previously agreed to by a *transmitter* and a *receiver*. The transmitter sends data to the receiver in a two-part message, sending a hypothesis, H, and then data coded on the assumption that the hypothesis is true, $D|H$. The hypothesis H stands for a set of models and thus gets a non-zero probability. There is a trade-off between the specificity of the set (equivalently the complexity of H) versus the fit of H to *expected* data. Note that not only continuous, but also discrete, parameters may be stated with less than maximum precision; we will see how this is useful for Bayesian networks.

Strict MML is computationally infeasible for all but the simplest problems [11], but practical, efficient approximations exist for many useful problems [12,13,14]. The work described here relies on stochastic MML approximations [3].

Minimum description length (MDL) inference was developed as an alternative to MML [15] and uses the same message length paradigm. MDL, however, favours universal priors and the selection of a model class rather than a parameterized model. A detailed comparison of MML and MDL has been given elsewhere by Baxter and Oliver[16].

BDe has its roots in Bayesian statistics. Like MDL, BDe attempts to find a model class rather than a parameterized model. BDe integrates over its prior on continuous parameters, whereas MML tries to segment continuous parameters into optimally sized regions and returns (a representative of) this region as an estimate.

3 Bayesian Networks

A Bayesian network is a directed acyclic graph (DAG) over a set of variables. Each node of the DAG represents a single variable. An arc commonly represents a direct causal relationship between a parent and a child. A node specifies the relationship between the node's variable and its parents, if any; we use conditional probability tables (CPTs), logit models and decision trees [8] for this.

An important concept when dealing with Bayesian nets is the 'statistical equivalence class' (SEC) [17]. Two DAGs in the same equivalence class can be parameterized to give an identical joint probability distribution; there is no way to distinguish between the two using only observational data over the given variables, although they may be distinguished given experimental data. Another important concept is the 'totally ordered model' (TOM). A TOM consists of a set of connections and a total ordering (permutation) of variables. Just as several DAGs may be in one SEC, several TOMs may realize a single DAG. TOMs are discussed further when introducing our MML coding scheme in §4.3.

4 Learning Global Structure

We briefly discuss the coding scheme for MDL and BDe and then build on these to present our MML coding scheme.

4.1 MDL Coding Scheme

We now summarize how Friedman encodes Bayesian networks using MDL. First the network structure is encoded, then the parameters of the network, and finally the data given the parameterized network. To encode the network structure, send the number of parents each node possesses, followed by the selection of parents out of all possible selections:

$$MDL_H = \sum_i \left(\log k + \log \binom{k}{|\pi(i)|} \right)$$

where k is the number of nodes and $\pi(i)$ is the parent set for node i. Friedman's code requires $\frac{1}{2} \log N$ nits per parameter to state a CPT (N is the sample size). The data requires

$$MDL_D = -\sum_{i=0}^{N} \log P(D_i)$$

Using this coding scheme and a heuristic search the DAG with the shortest description length is accepted as the best model.

4.2 BDe Scheme

Heckerman et al.'s BDe metric [6], based on a previous Bayesian metric of Cooper and Herskovits [18], has been augmented with decision trees by Friedman [1] whose implementation is used for comparison in section 7. The suggested prior is based on an edit distance from an expert supplied network. Friedman uses a prior based on the MDL prior outlined above with $P(H) \propto 2^{MDL_H}$.

Once the network structure has been stated, we must integrate

$$P(D|H) = \int P(D|\theta, H) P(\theta|H) d\theta$$

where $P(\theta|H)$ is the prior parameter density and $P(D|\theta, H)$ is the probability of the data given the parameterized network. Using a Dirichlet prior, the closed form solution is:

$$P(D|H) = \prod_i \prod_{pa_i} \frac{\Gamma(\sum_{x_i} N'_{x_i|pa_i})}{\Gamma(\sum_{x_i} N'_{x_i|pa_i} + N(pa_i))} \times \prod_{x_i} \frac{\Gamma(N'_{x_i|pa_i} + N(x_i, pa_i))}{\Gamma(N'_{x_i|pa_i})}$$

where x_i is a node instantiation, pa_i an instantiation of $\pi(i)$, $\Gamma(x)$ is the Gamma function and $N(\cdot)$ counts the number of sample cases matching an instantiation. Heckerman's default "equivalent sample size", $N' = 5$, is used.

4.3 MML Coding Scheme

Whereas many methods use a uniform prior over SECs or DAGs, CaMML uses uniform priors over totally ordered models (TOMs). In essence, TOMs are DAGs

with a consistent total order selected. A TOM can be thought of as a way of realizing its DAG; in effect, TOMs specify distinct possible worlds in which the DAG is true. We should prefer to employ non-informed, uniform (maximum entropy) priors only as a last resort — i.e., when we arrive at the most primitive level of description, in this case TOMs rather than DAGs or SECs. (For more discussion of this approach see Korb and Nicholson, Chapter 8. [4].)

CaMML's stochastic search algorithm (section 4.4) samples TOMs, but it counts DAGs, so that each TOM contributes probability mass to its DAG. The algorithm has the following stages. A TOM is sampled in the MCMC process. The corresponding DAG is "cleaned" by deleting weak arcs whenever this reduces the total message length. The clean DAG is counted, as is its SEC. Repeated counting allows us to estimate the posterior distribution over the DAG and SEC space.

When the sampling phase is over, SECs are also grouped in case the data does not justify choosing between them, using a Kullback-Leibler (KL) divergence test. As with lower level groupings, such a group may gain enough probability mass to be preferred even when the single SEC might not.

The encoding of a single TOM has two parts: a list of arcs and a total ordering. The arcs can be encoded in $m \times \log P_a + \left(\frac{k(k-1)}{2} - m\right) \times \log(1 - P_a)$ nits. Where k is the number of nodes in the network, m is the number of arcs present and P_a is the prior probability of arc presence (default 0.5). The cost to state the total ordering is simply $\log k!$ nits. In addition to the TOM we must also use $\log p(data|TOM)$ nits to express the data — see section 5. This scheme forms the basis of an efficient code employing our prior beliefs about network structure.

4.4 The MCMC Search

An MCMC search algorithm allows us to approximate a posterior distribution over DAGs and SECs by sampling TOM space. The algorithm used follows:

1. Simulated Annealing to find the best single TOM. This optimal TOM is used to estimate P_a and provide a starting position for our sampling.
2. Attempt a mutation on the current TOM M transforming it into M'
   (a) **Temporal:** Swap the order of two neighbouring nodes in the total ordering. If an arc exists, reverse its direction.
   (b) **Skeletal:** Select two nodes at random and toggle the existence of an arc between them.
   (c) **DoubleSkeletal:** Select three nodes at random, toggle the arcs from the first two to the final (in the total ordering).
   (d) **ParentSwap:** Select three nodes such that $a \rightarrow c$ but not $b \rightarrow c$ [1] and toggle the arcs ac and bc. This effectively removes one parent and replaces it with another.

   Mutations c and d are not strictly necessary since they are compositions of the other mutations, but speed up the sampling process.

[1] If not possible, choose a different mutation.

3. Accept M' as the new sampled M if

$$\frac{\log P_{MML}(M) - \log P_{MML}(M')}{temperature} > \log U[0,1]$$

else retain M. Sampling is conducted at $temperature = 1.8$. [2]
4. Add a weight of $\frac{\exp^{P_{MML}} - \exp^{P_{BestMML}}}{temperature}$ to the current TOM's clean representative DAG and SECs accumulated weight. $P_{BestMML}$ is a common factor to avoid underflow.
5. Loop to 2 until a set number of steps are complete.

Step 4 above refers to a TOM's clean representative DAG and SEC; mutation continues from the current "unclean" TOM at the next iteration.

5 Learning Local Structure

The coding scheme and the MCMC algorithm place few requirements on the method used to represent local structure (i.e., parent-child relationships). The present work uses CPTs, decision trees, logit and hybrid models. The later is able to choose between other model types on a node-by-node basis. Here we describe the addition of local structure to our MML metric; see Friedman [1] for more detail on MDL and BDe.

5.1 CPTs

The CPT is the standard building block of Bayesian networks. If a child (variable) takes one of S possible values (states), a multi-state distribution having $S-1$ parameters must be specified for each combination of parent values. Wallace and Boulton [12] gave the MML calculations for the multi-state distribution. The message length was shown to be equivalent to using an adaptive code for the data with an extra "penalty" of a fraction of a nit, $\frac{1}{2}\log\frac{\pi e}{6} \approx 0.176$ per parameter; the consequence of using an estimate with optimum precision [13]:

$$mesglen = \frac{|pa| \times (|x|-1)}{2}\log\frac{\pi e}{6} + \sum_{pa_i}^{|pa|}\log\left(\frac{(N(pa_i)+|x|-1)!}{(|x|-1)! \times \prod_{x_i}^{|x|}(N(pa_i,x_i)!)}\right)$$

where $|pa|$ and $|x|$ are the number of parent and child states respectively.

5.2 Decision Trees

To code decision trees we begin with Wallace and Patrick's code [8]. Briefly, each leaf or split node has an initial cost of 1 bit. Also stated for each split

[2] A high temperature causes TOM space to be more widely traversed, low temperature makes the sampling more likely to stay near the original model. A temperature of 1.8 was used throughout this work.

is the variable being split; an attribute cannot be "reused" and so this costs $\log k - depth$ nits. Each leaf must state a model for that node and also the data given that model. Multinomial models are the natural choice here, as with CPTs.

In general, decision trees can ignore parents that are irrelevant by never splitting on them. However, in the context of local structure in a Bayesian network, we require that all parents should be used, since the coding of the network structure implies it. To achieve this, we correct the message length and force our trees to split on each parent at least once, other trees being disallowed. Selecting parents is thus the responsibility of network topology discovery, rather than the local node encoding.

Our strategy for reclaiming lost probability (from the disallowed trees) is based on the number of trees with n split nodes, i.e., on the Catalan numbers defined as $cat(n) = \binom{2n}{n} \div (n+1)$. The prior probability of a given tree structure with n splits can be calculated as $p_n = 2^{-(2n+1)}$; by multiplying these numbers we calculate the total proportion of prior allocated to models with n splits. As can be seen in Table 1, this prior is skewed towards models with few splits, as one would expect.

Table 1. Catalan numbers

n	cat(n)	$2^{-(2n+1)}$	$p_n cat(n)$	$\sum_{i=0}^{n} p_i cat(i)$
0	1	0.50000	0.50000	0.50000
1	1	0.12500	0.12500	0.62500
2	2	0.03125	0.06250	0.68750
3	5	0.00781	0.03906	0.72656
4	14	0.00195	0.02734	0.75391
5	42	0.00048	0.02051	0.77441
6	132	0.00012	0.01611	0.79053
7	429	0.00012	0.01309	0.80362
∞	∞	0		1

Table 2. Savings made(in bits): where n = number of parents, min = minimum number of splits, max = maximum number of splits, s = number of splits made

n	min/max	$P_{s<min}$	$P_{s>max}$	$P_{invalid}$	saving
0	0...0	0.000	0.500	0.500	1.00
1	1...1	0.500	0.375	0.125	3.00
2	2...3	0.625	0.273	0.102	3.30
3	3...7	0.688	0.196	0.116	3.11
4	4...15	0.727	0.140	0.133	2.91
5	5...31	0.754	0.099	0.147	2.77

Taking the simplest example of a leaf with no parents, it is obvious that our tree structure will be that of a single leaf. However, under our original prior 1 *bit* is still required to express this structure. For a (binary) tree with 1 parent the original structure cost would be 3 $bits$, one for each split and one for each leaf. In general, it would be reasonable to pay this cost as a decision to split is actually made, but in the present context the modified priors tell us that these splits are always required, so the penalty should be removed. A slightly more complex case arises for two or more parents where our priors allow between N and 2^N splits where N is the number of parents. An approximation of this saving is used for n-ary variables.

To further illustrate, we take a tree with three binary parents. We are constrained to have at least three splits, and no more than seven splits. From Table 1 we see that trees with less than three splits use 0.688 and trees with more than seven use $1-0.804 = 0.196$ of our prior hypothesis space. So our original prior has 0.884 wasted on impossible hypotheses! By subtracting $-\log 1-0.884 = 3.11$ $bits$

from our message length we effectively redistribute the probability mass from trees with an invalid number of splits to those with appropriate splits.

By examining the table of Catalans, Table 1, it is possible to calculate the saving, as seen in Table 2. Using our true prior to calculate Decision Tree costs is obviously a good thing to do, but it becomes especially important when dealing with hybrid models so that CPTs and trees can compete fairly.

5.3 Logit

We follow Neil et al.[2] in supporting MML logit models of local structure. A CPT treats each parameter independently. It is common, however, for there to be local structure, with some parameters dependent upon others. A first order logit model allows us to exploit this.

$$P(X = x|Z_1 = z_1, Z_2 = z_2, \ldots Z_n = z_n) = \frac{e^{a_i + b_{iz_1} + c_{iz_2} \cdots}}{\sum_{j=1}^{|X|} e^{a_j + b_{jz_1} + c_{jz_2} \cdots}}$$

When the effect of each parent is independent of the effects of other parents (so joint parameters are not independent), we expect our logit model to give a better representation of a distribution than a CPT would; given interacting parents a CPT or DTree would be expected to perform better. To get the best of both worlds, hybrid models are useful.

5.4 Hybrid Models

Hybrid models, as the name suggests, are combinations of two or more models of local structure; in this case we have combined CPTs, decision trees and logit models. Previous work [2] combined CPTs and logit models with some success.

We define a hybrid model as a model which can choose between competing local models to give the best result. The search costs a decision tree, a logit and a CPT, choosing the one with the lower MML cost. It is also necessary to add $\log 3$ *nits* to the node's cost — treating models as being equally likely apriori. This extra cost is not required for models with zero or one parent, as the CPT, DTree and logit have equivalent expressive power, so a CPT is used.

6 Evaluation

Our results are defined in terms of KL divergence from a true model (when known) or log probability on a test set (otherwise). Tests were also run on data generated from artificial networks having varying degrees of "decision treeness" in their local structure to show cases where CPTs should be favoured and where decision trees should be favoured; hybrid models should perform well across the whole spectrum. This degree is quantified by P_l: the number of leaves a decision tree will possess is at least $P_l \times |pa|$. Low values correspond to more local structure. Our simple tree generation algorithm follows:

1. Begin with an empty tree (single leaf)
2. Choose a leaf at random and split using any unused variables
3. If less than $P_l \times |pa|$ leaves exist, goto 2
4. For each variable has not been split on, choose a leaf and split on it.

In addition to the random networks we use the "insurance" network, which consists of 27 variables with arity ranging from 2 to 5, and also the real datasets listed in Table 3.

Error bars are shown at 1.96 standard errors (SE) from the mean. Ideally, pairwise comparisons over all searches would have been used, but this becomes unmanageable when comparing 8 unique metrics.

7 Results

Figure 2 shows results for artificial networks with varying degrees of local structure, i.e., varying P_l. The KL divergence of the inferred network from the true network is plotted. Results are averaged over 100 trials. CaMML performs much better than BDe and MDL, both with CPTs and DTrees. As expected, CaMML DTrees do best when P_l is low and CPTs do better when P_l is high. The CPT-DTree hybrid model does well across the range.

Logit models were excluded in this test, as their assumption of a non-interactive distribution is not met here and as such would perform poorly. An experiment similar to this (although varying levels of first and second order effect strength) compares CPTs, logits and CPT-logit hybrid models is found in Neil et al[2]. That paper shows logit models outperforming CPTs when their assumptions are warranted and CPTs winning when they are not. Again, hybrid models perform well throughout.

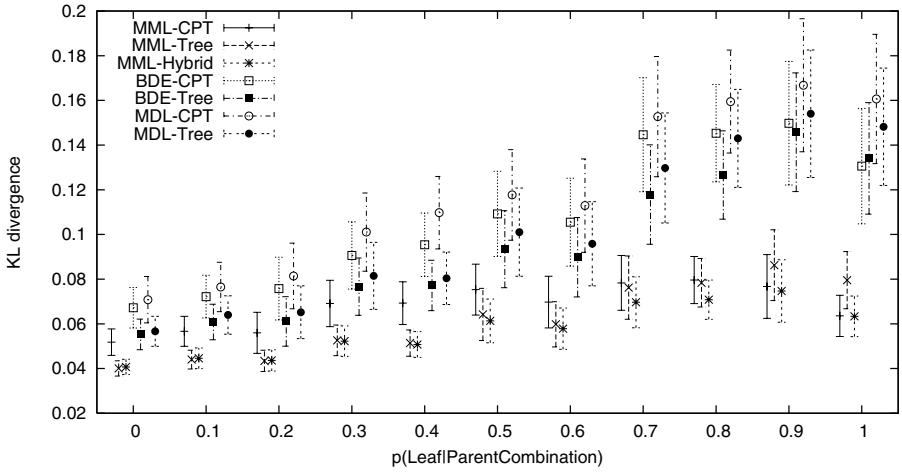


Fig. 2. Varying P_l, $P_a = 0.25$, N = 1000, 7 nodes, arity = [2,5,10,7,3,3,2] 100 folds. Error bars at 1.96 SE.

Figure 3 shows KL divergence results for the insurance network while varying the training sample size. Plots were adjusted by multiplying KL values by $N/\log N$, as suggested in [1], keeping values approximately constant across the graph by compensating for KL asymptoting to zero as N grows large. As in Figure 2, MDL performs worse than MML and BDe across the board and as such has been removed to reduce clutter.

It is clear that for both MML and BDe (and unshown MDL results) that tree based learners often outperform CPT learners for this example. The difference is especially evident for large datasets. This result confirms Friedman's work.

Of more interest, however, is the comparison to logit based learners. Our MML logit learner significantly outperforms all out non-logit learners for small datasets ($N < 1000$) but performs much worse for large datasets. This is due to logit models being able to approximate first order effects better than CPTs or trees for small datasets, but being unable to express second (and higher) order effects when enough data is given to reveal them. Our MML hybrid learner (using CPT, tree and logit models) has the best of both worlds. It performs as well as the logit model for small sample sizes, then roughly as well as the tree based model for larger data sizes.

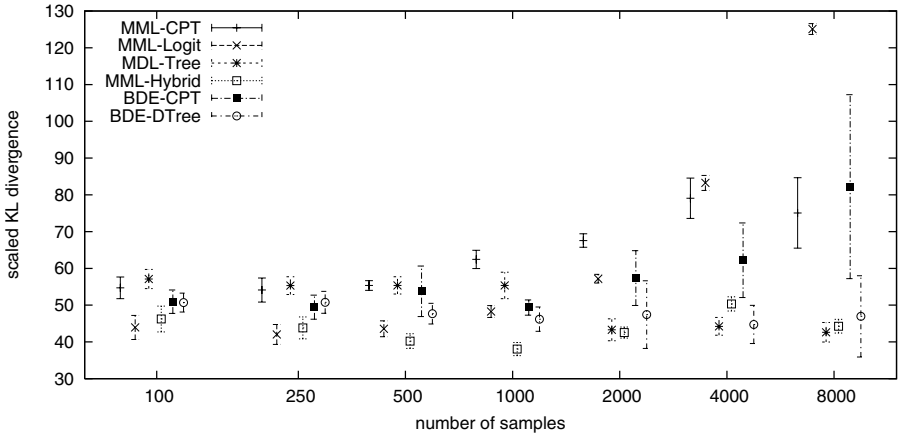


Fig. 3. Insurance Network, 10 folds. Error bars at 1.96 SE

Figure 4 shows results from several real datasets summarized in Table 3. Once again MDL performed badly and is not shown. Eight datasets are examined including the six used by Neil [2] and two larger datasets where we would expect decision trees to do well.

Table 3. Real Datasets from UCI repository

name	description	k	arity	N
Zoo	Animal attributes	17	2-7	101
ICU	Intensive care unit	17	2-3	200
Flare	Solar flares	13	2-7	323
Voting	US congress votes	17	2-3	435
Popularity	Childhood popularity.	11	2-9	478
kr-vs-kp	Chess end games	37	2-3	3196
Mushroom	Poison mushrooms	23	2-12	8124
Nursery	Child care.	9	2-5	12960

Comparing CPT and tree based learners (for MML and BDe) we see comparable results for small datasets, with tree learners performing significantly better on larger datasets. To visually clarify results, all $logP$ values have been normalized by the worst scoring learner for each dataset.

In all but one of our datasets there is a clear winner between MML Tree and logit models, with logit winning on small datasets, but loosing badly on large datasets. Our hybrid model does well throughout having several significant wins against each rival MML learner, but no significant losses.

In results not shown, all hybrid combinations of models were examined. That is a "CPT-Logit", "CPT-Tree" and "Tree-Logit", in addition to the "CPT-Tree-Logit" shown. It was evident that when our hybrid model contained a logit component, it did well on small datasets and when it contained a tree component it did well on large datasets. Hybrid learners with both options did well on small and large datasets. Removing CPTs from the "CPT-Tree-Logit" learner has a much smaller effect than removing either the tree or logit component.

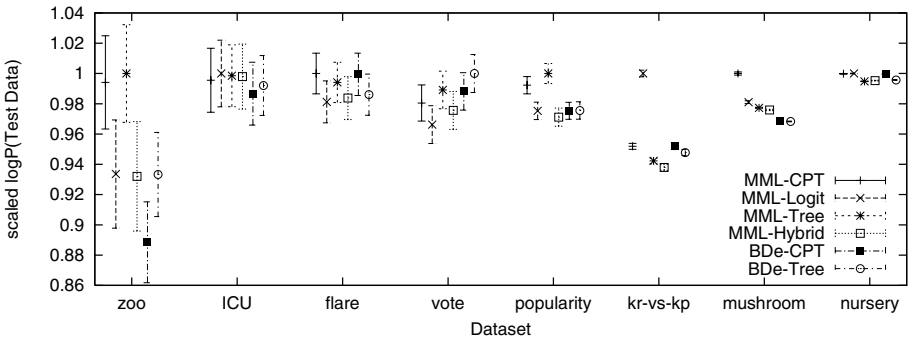


Fig. 4. UCI Repository datasets, 100 folds. Error bars at 1.96 SE

8 Conclusion

We have shown that trees, logit and hybrid models of local structure can be introduced successfully into the CaMML search procedure. When jointly incorporated in hybrid model discovery, the result is a flexible learning procedure which automatically accommodates data set sizes by preferring simple local structure representations (e.g., logit models) given small data sets and by finding richer representations (decision trees and/or CPTs) given large data sets. We also compared these MML metrics with BDe and MDL metrics, finding that generally BDe and CaMML do well, with MDL performing poorly.

Acknowledgments

We would like to acknowledge the late Chris Wallace whose work on MML, Bayesian networks and decision trees was pioneering.

References

1. Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In: Uncertainty in Artificial Intelligence. (1996)
2. Neil, J.R., Wallace, C.S., Korb, K.B.: Learning Bayesian networks with restricted causal interactions. In: Uncertainty in Artificial Intelligence. (1999)
3. Wallace, C.S., Korb, K.B.: Learning linear causal models by MML samplling. In Gammerman, A., ed.: Causal Models and Intelligent Data Management. Springer-Verlag (1999)
4. Korb, K., Nicholson, A.: Bayesian Artificial Intelligence. CRC Press (2003)
5. O'Donnell, R.T., Nicholson, A.E., Han, B., Korb, K.B., Alam, M.J., Hope, L.R.: Causal discovery with prior information. 19th Australian Joint Conf on AI (2006)
6. Heckerman, D., Geiger, D., Chickering, D.: Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning **20**(197-243) (1995)
7. Lam, W., Bacchus, F.: Learning Bayesian belief networks. Computational Intelligence **10** (1994)
8. Wallace, C., Patrick, J.: Coding decision trees. Machine Learning **11** (1993) 7
9. Bayes, T.: An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Soc. of London (1764/1958) reprinted in Biometrika 45(3/4) 293-315 Dec 1958.
10. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3) (1948) 379–423
11. Farr, G.E., Wallace, C.S.: The complexity of strict minimum message length inference. Computer Journal **45**(3) (2002) 285–292
12. Wallace, C.S., Boulton, D.M.: An information measure for classification. The Computer Journal **11** (1968) 185–194
13. Wallace, C.S.: Statistical and Inductive Inference by Minimum Message Length. Springer, Berlin, Germany (2005)
14. Allison, L.: Models for machine learning and data mining in functional programming. Journal of Functional Programming (2005)
15. Rissanen, J.: Modeling by shortest data description. Automatica **14** (1978)
16. Baxter, R., Oliver, J.: MDL and MML: similarities and differences. Technical Report 207, Dept of Computer Science, Monash University (1994)
17. Chickering, D.: A transformational characterization of equivalent Bayesian network structures. In: Uncertainty in Artificial Intelligence. (1995)
18. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning **9** (1992) 309–347