UAI2002 DAVIES & MOORE 119

# Interpolating Conditional Density Trees

Scott Davies and Andrew Moore School of Computer Science Carnegie Mellon University

Pittsburgh, PA 15213 [scottd, awm]@cs.cmu.edu

# Abstract

Joint distributions over many variables are frequently modeled by decomposing them into products of simpler, lower-dimensional conditional distributions, such as in sparsely connected Bayesian networks. However, au­ tomatically learning such models *can* be very computationally expensive when there are many datapoints and many continuous vari­ ables with complex nonlinear relationships, particularly when no good ways of decom­

posing the joint distribution are known *a pri­*

*ori.* In such situations, previous research has

generally focused on the use of discretization techniques in which each continuous vari­

# 1 INTRODUCTION

Bayesian networks are a popular method for represent­ ing joint probability distributions over many variables. A Bayesian network contains a directed acyclic graph G with one vertex V; in the graph for each variable X; in the domain. The directed edges in the graph specify a set of independence relationships between the vari­

ables. Define IT; to be the set of variables whose nodes in the graph are "parents" of V;. The set of inde­

pendence relationships specified by G is then as fol­ lows: given the values of IT; but no other information, X; is conditionally independent of all variables corre­

sponding to nodes that are not V; 's descendants in the

graph. These independence relationships allows us to

decompose the joint probability distribution P(X) as

�

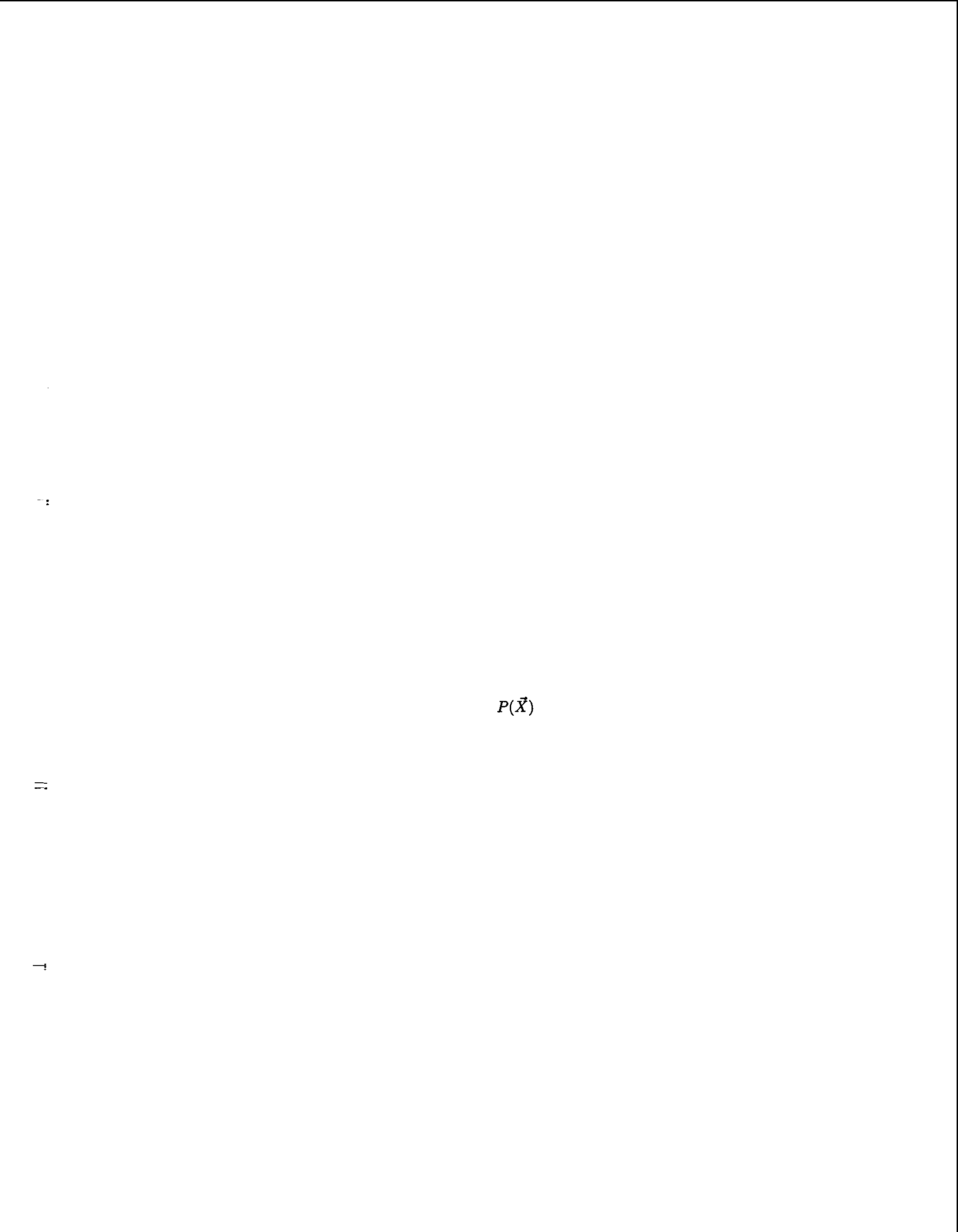
able has a single discretization that is used

*N*

= I1i=l

P(X;III;), where *N* is the number of

throughout the entire network.



In this paper, we present and compare a wide variety of tree-based algorithms for learning and evaluating conditional density estimates over continuous variables. These trees can be thought of as discretizations that vary ac­ cording to the particular interactions being modeled; however, the density within a given leaf of the tree need not be asumed con­ stant, and we show that such nonuniform leaf densities lead to more accurate density esti­ mation. We have developed Bayesian net­ work structure-learning algorithms that em­ ploy these tree-based conditional density rep­ resentations, and we show that they can be used to practically learn complex joint prob­ ability models over dozens of continuous vari­ ables from thousands of datapoints. We focus on finding models that are simultaneously ac­ curate, fast to learn, and fast to evaluate once they are learned.

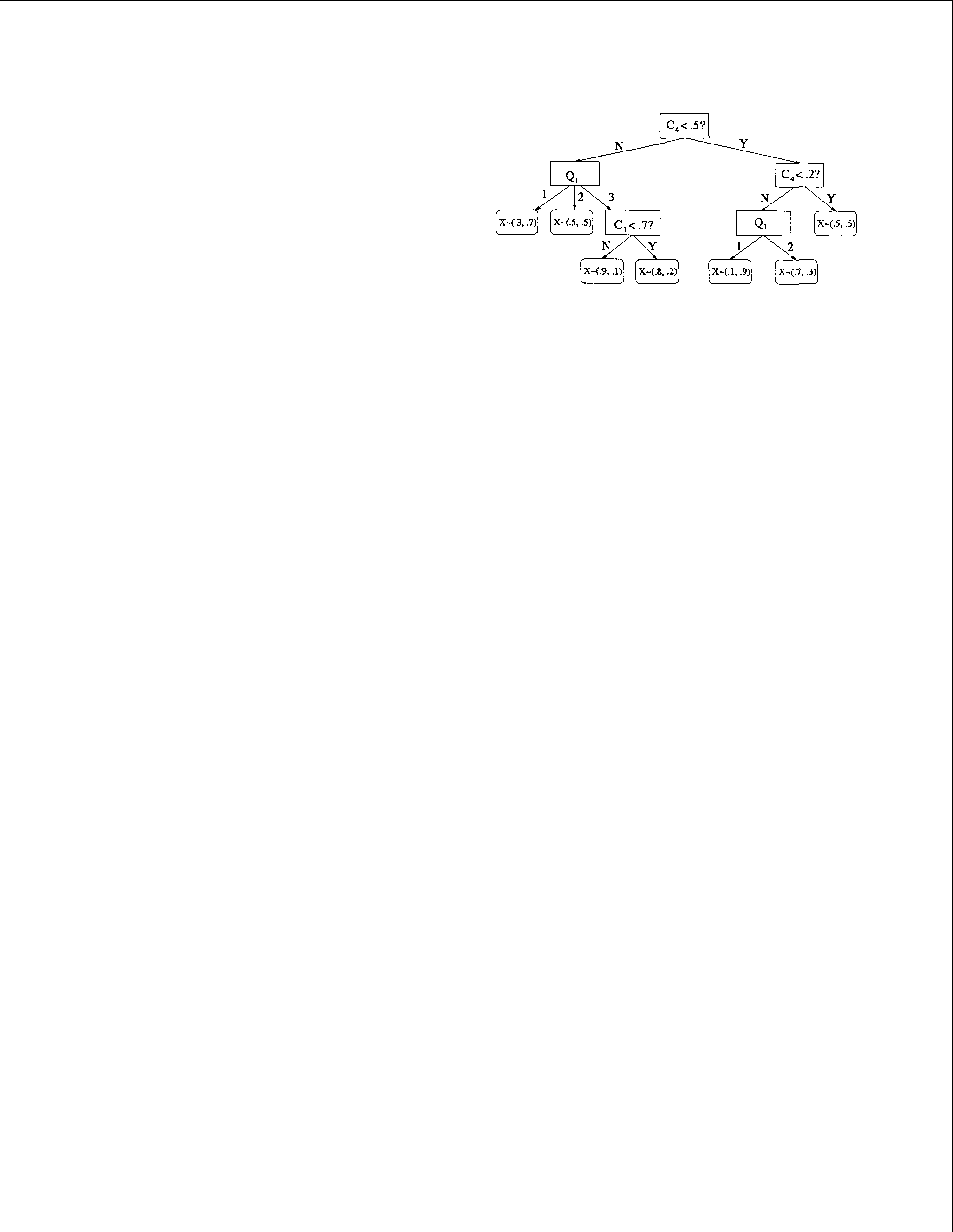
variables in the domain. Thus, if in addition to G we also specify P(X;IIT;) for every variable X;, then we have specified a valid probability distribution P(X) over the entire domain.

Bayesian networks are most commonly used in sit­ uations where all the variables are discrete; if con­ tinuous variables are modeled at all, they are typ­ ically assumed to follow simple parametric distri­ butions such as Gaussians (e.g. (Heckerman and Geiger, 1995)). Some researchers have recently in­ vestigated the use of complex continuous distributions within Bayesian networks; for example, weighted sums of Gaussians (Driver and Morrell, 1995), Gaussian kernel-based density estimators (Hofmann and Tresp, 1995), and Gaussian processes (Friedman and Nach­ man, 2000) have been used to approximate conditional probability density functions. Such complex distribu­ tions over continuous variables are usually quite com­ putationally expensive to learn. This expense may not be too problematic if an appropriate Bayesian network structure is known beforelland. On the other hand, if the dependencies between variables are not known *a*

*priori* and the structure must be learned from data'

120 DAVIES & MOORE UAI2002

## then the number of conditional distributions that must be learned and tested while a structure-learning algo­ rithm searches for a good network can become unman­ ageably large.



In such cases, the search over network structures is usually performed using a discretized version of the data, where the range of each variable is divided into some number of bins and all values of a given variable within a given bin are considered equiva­ lent. This discretization can performed once before network structure-learning, and the resulting network structure can then be reparameterized with continu­ ous distributions in a final step ((Monti and Cooper, 1998b), (Monti and Cooper, 1999)); or, a simultane­ ous search of both network structures and discretiza­ tion policies can be performed ((Friedman and Gold­ szmidt, 1996a), (Monti and Cooper, 1998a)). In this previous research, however, the discretization of each variable has been *global* - that is, the same discretiza­ tion for any particular variable is employed for all the interactions in which it is involved.

Decision trees (see e.g. (Quinlan, 1986), (Breiman et al., 1984)) have been used previously in Bayesian networks over discrete variables (Friedman and Gold­ szmidt, 1996b) in cases where full conditional contin­ gency tables could be too large to learn accurately from limited data. In this paper, we propose and eval­ uate four different tree-based approaches to the condi­ tional density estimation of continuous variables, with different tradeoffs between accuracy, learning speed, and evaluation speed:

* CART(Breiman et a!., 1984)-like trees, which are fast to learn and evaluate but are inadequate for accurately representing complex conditional dis­ tributions.
* *Stratified conditional density trees,* which are more computationally expensive to learn than CART-like trees but are still fast to evaluate and are better than CART-like trees at general­ purpose density estimation.
* Joint density trees that are used conditionally. These are fast to learn, and (somewhat surpris­ ingly) appear more accurate than stratified condi­ tional density trees. Unfortunately, they are com­ putationally expensive to evaluate.
* *Approximately conditionalized joint density trees,* which combine the best features of the previous three tree types in that they are fast to learn, fast to evaluate, and accurate at density estimation.

## In Sections 2.1- 2.7, we explain, compare and con­ trast these four different types of trees, and provide

Figure 1: An example of a conditional density tree (or classification tree) for predicting the distribution of a binary variable X as a function of several other variables.

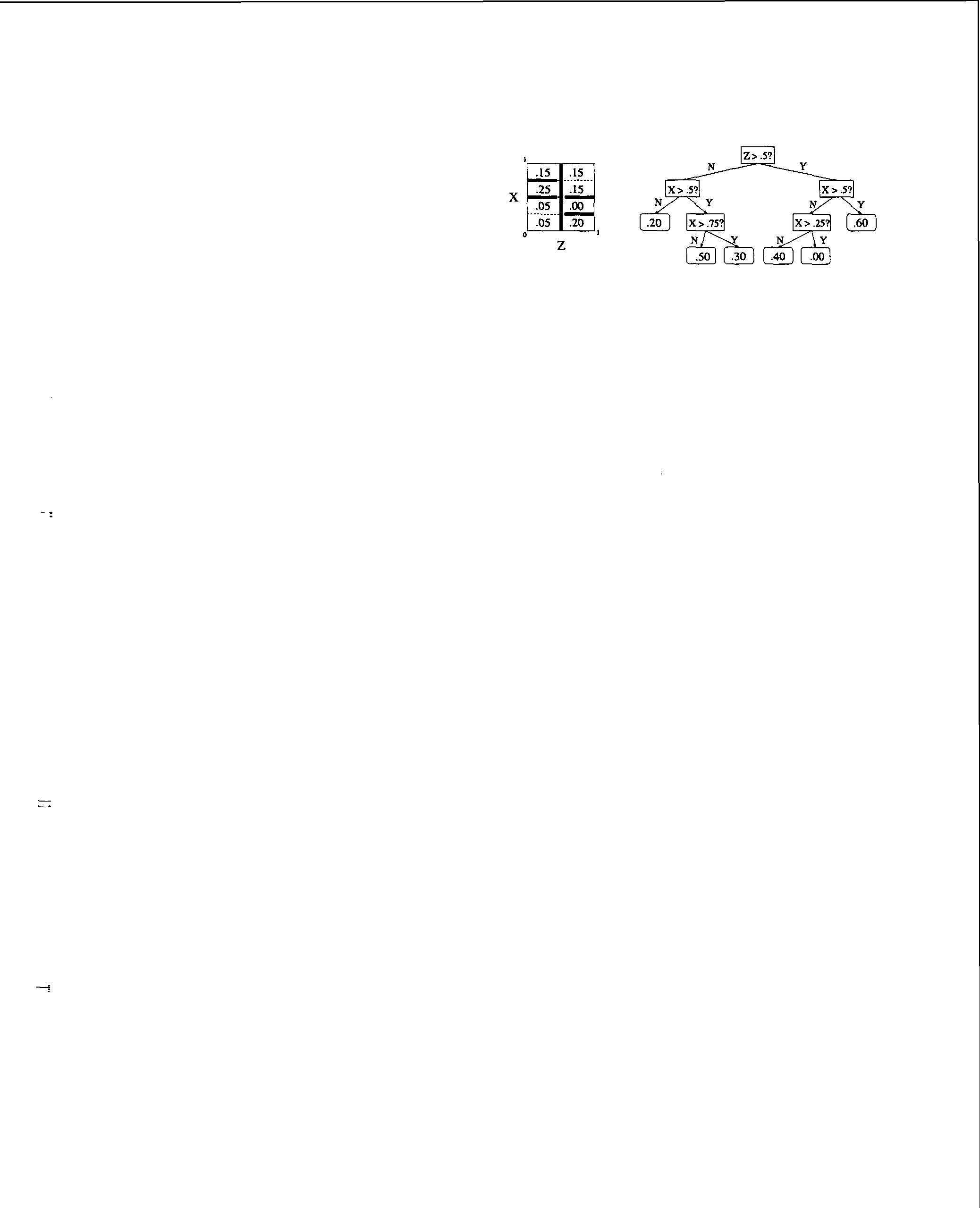
experimental results on real and synthetic datasets in which we keep the set of conditional distributions be­ ing modeled constant. In Section 2.8, we briefly dis­ cuss a Bayesian network structure-learning algorithm that employs these trees, and show that the resulting overall algorithm can practically find accurate factored models that are also fast to evaluate, compared to global mixture model-learning algorithms such as Au­ toClass. Finally, in Section 3 we summarize our find­ ings and discuss possible avenues for future research.

1. TREE AND LEAF TYPES
   1. Classiftcation and re ression trees

## Figure 1 shows an example decision tree in which the distribution of a. binary variable X is predicted as a function of several other variables, some of which are discrete (the Q's) and some of which are continuous (the C's). To find the distribution of X, the predic­ tion algorithm simply starts at the root of the tree (shown at the top of our diagram) and follows a path down the tree's branches according to the values of the other variables until it reaches a leaf. For exam­ ple, if the continuous variable c4 is less than .5, and the trinary discrete variable Q1 has a value of 1, then the algorithm would predict that X has a 30% chance of taking on its first possible value and a 70% chance of its second. Such decision trees for predicting the distributions of discrete variables are also known as "classification trees" in contexts where the task ulti­ mately involves guessing a single value (typically the most likely value) for the variable being predicted.

Regression trees (e.g. (Breiman et al., 1984)) have structures similar to those of decision trees, but the leaves of these trees provide information about the dis­ tributions of a continuous variable X instead. Typi­ cally in regression this information is restricted to a point estimate of the variable's mean; this mean may

UAI2002 DAVIES & MOORE 121



be constant, or it may be (for example) a linear func­ tion of the parent variables. In order to obtain an actual density estimate, a variance can be supplied as well as the mean in order to specify the parameters of a Gaussian.

Decision and regression trees are typically learned by greedy top-down divide-and-conquer algorithms; *we* employ such an algorithm in the experiments described in this paper. A decision or regression "stump" of depth one is grown for each possible branching vari­ able. The algorithm then chooses the branch vari­ able whose corresponding stump most increased the total conditional log-likelihoods of a randomly selected subset of the training data that was held out during the stump-training process. The algorithm then re­ cursively learns the branch node's children using the appropriate subsets of the training data. When split­ ting on a discrete variable, the resulting branch always has one child for every possible value of the branch variable; when splitting on a continuous variable, the branch has two children corresponding to whether the variable is � or *>* the midpoint of the current pos­ sible subrange for that variable. (The algorithm is initially provided with a hypercube over the contin­ uous variables in which al nonzero probability is sumed to lie.) Braftching is terminated when fewer thaft ten training datapoints are consistent with the current subtree. A separate random holdout set of the training data is then used to prune the learned deci­ sion tree. Many variations of this learning algorithm are considered in the full version of this paper (Davies,

## 2002).

Regression trees may be adequate for representing con­ tinuous conditional distributiofts in situatiofts where they are in fact near-Gaussian, or when the problem involves guessing a point estimate and then being pe­ nalized by its squared distance to the real value. How­ ever, there are other situations in which we may wish to have reasonably accurate models of distributions that are more complicated, e.g. multimodal. There are many possible criteria to use when judging the ac­ curacy of such models; one of the most common is the Kullback-Leibler divergence of the model from the true distribution. Since we will be learning models from scientific data with unknown true distributions in our evaluations, we will use the log-likelihoods of test sets in cross-validation experiments as empirical aftalogues of the KL divergence.

* 1. Stratified conditional density trees

There is no reason in principle to stop at a simple parametric distribution for the child variable once the branching on pareftt (i.e. "input") variables has fin­ ished. Instead, one can employ a *stratified conditional*

Figure 2: A stratified conditional density tree.

*density tree* in which any path from the root of the tree to a leaf first passes through a sequence of branch nodes that only test the parent variables, and then through aftother sequence of branch nodes that only test the child (i.e. "output") variables. Such a tree for estimating the conditional density of one contin­ uous variable X given another Z might look like the one in Figure 2, where for clarity *we* have listed the conditional probability *masses* inside the leaves rather than the conditional probability deftsities; the densi­ ties are trivially computed from these masses by di­ viding by the volumes of the leaves. Note that in or­ der to represent a valid conditional distribution, the masses in any subtree containing no branches on the parent variables must sum to 1. This constraint is what forces us to learn trees with this stratified branch­ ing structure: if branches on the input and output vari­ ables are allowed to alternate arbitrarily, then the con­ straint becomes nonlocal, making divide-and-conquer approaches to learning the tree difficult. (See (Davies, 2002) for details.) The recursive algorithm we employ to learn stratified conditional density trees is identical to the algorithm we *use* to learn decision and regression trees, except that wherever the decision or regression tree learner would call a routine that returns a simple leaf distribution fitting the provided data, the strat­ ified conditional density tree learner calls a subtree­ learning procedure. This subtree-learning procedure is also idefttical to the algorithm we use to learn deci­ sion and regression trees, except the subtrees it learns branch only on the output variable, and at each branch the algorithm divides the total conditional probabil­ ity mass that each child is allocated according to the proportion of datapoints that fall in that child's sub­ tree. Because an entire subtree is learned where a CART-like algorithm merely needs to learn a leaf dis­ tribution, learning stratified conditional density trees in this maftner is significantly more computationally expensive to learn than CART-like trees. However, as we shall see, they *can* provide much more accurate density estimation.

Given the total conditional probability mass that lies a given leaf, we are still left with a choice of how to distribute it within that leaf. If the mass within

122 DAVIES & MOORE UAI2002

each leaf is distributed with uniform density, then the stratified conditional density tree is essentially employ­ ing variable-resolution histograms in place of the sim­ ple parametric distributions (such as Gaussians) that CART-like trees use. However, most other choices (as

mentioned in Section 2.5) lead to more accurate den­

sity estimation.

* 1. Usin joint density trees conditionally

While the stratified conditional density trees discussed in the previous section can model conditional density trees much more accurately than CART-like single­ level conditional density trees, they are computation­ ally expensive to learn. There are many heuristics that could be tried to alleviate this problem, such as learn­

ing a CART-like tree first and using this tree's struc­

ture as a starting point for a stratified conditional den­ sity tree. However, such heuristics would be unlikely to increase the accuracy of the resulting models, and likely to decrease it. As it turns out, it is possible to achieve more accurate density estimation *and* faster learning using an alternative approach.

In this section we discuss the use of density trees modeling joint distributions P(X;, IT;) to obtain condi­ tional density estimates P(X;jiT;). Each leaf l of such a tree specifies a joint probability P(X;,IT;jl): that is, the probability that X; and IT; take on specific values

within the leaf's range, given that the datapoint lies somewhere within the bounds of l. Assuming we have a density tree representing P(X;, IT;), we can obtain an estimate for a particular P(x;j1ri) as follows:

# P(x;j1ri) = L P(lj1ri) · P(x;Jlfi,l)

particular branch node, and the joint log-likelihoods of all the variables in {X;} *U* IT; are used for evaluat­ ing any particular branch choice rather than just the conditional log-likelihood of X;.

Joint density trees are trivially capable of represent­ ing Bayesian classifiers when used conditionally in this manner. In particular, since each leaf in the density trees employed in this paper models each discrete vari­ able independently of all other variables (using a multi­ nomial distribution), a Naive Bayes classifier for dis­ crete variables is obtained in the special case where the density tree is a one-level density stump with a root node branching on the variable to be predicted. Such Naive Bayes classifiers have previously been used

to model the conditional distributions within Bayesian networks (Heckerman and Meek, 1997). A commonly used Bayesian classifier for continuous variables is to

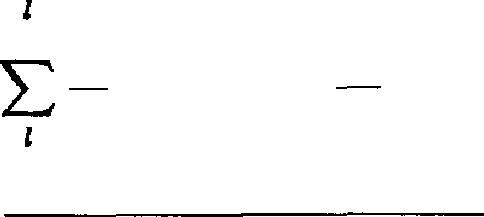
model each class distribution with a Gaussian; this clasifier is obtained simply with a density stump branching on the class variable with leaves employ­ ing Gaussian distributions over the continuous vari­ ables. More generally, suppose a joint density tree over discrete variables has a branch structure similar to the branch structure of a stratified conditional den­ sity tree: that is, once the output variable is tested in a branch node, no further tests can be performed on the input variables in subsequent levels of the tree. When this joint density tree is used to estimate condi­ tional distributions for the output variable, it is sim­

ilar in form and function to a hybrid decision tree I

Naive Bayesian classifier also developed in previous re­

search (Kohavi, 1996). In the most general case when the tree has an arbitrary branch structure (and the variables are not necessarily discrete), the algorithm

for computing conditional distributions essentially cre­

P(l) · P(lfiil)

=

L:z• P(l') . P(lfill)

. P(x;j7r�;,l)

ates a Bayesian classifier "on the fly" across different parts of the tree to determine which of the leaves con­

# P(lc) · P(lfillc) · P(x;jlfi, lc)

=

L:z• P(l') · P(lfill')

where the summation over l collapses to a single leaf

lc consistent with both x; and 1ri, since all other leaves l have either P(lfiil) or P(x;jlfi,l) equal to zero. This equation gives us a simple way of calculating con­

ditional distributions P(X;jiT;) from trees modeling joint distributions P(X;, IT;), assuming the distribu­ tion P(X;, IT;jl) within each leaf l can be marginalized to compute P(IT;jL) and conditionalized to compute

# P(X;jiT;, L).

The algorithm we use to learn joint density trees of this form is identical to the learning algorithm we use

for decision I regression trees, except the joint density

tree learning algorithm treats X; and fi on equal foot­ ing: either X; or a variable in IT; can be tested at any

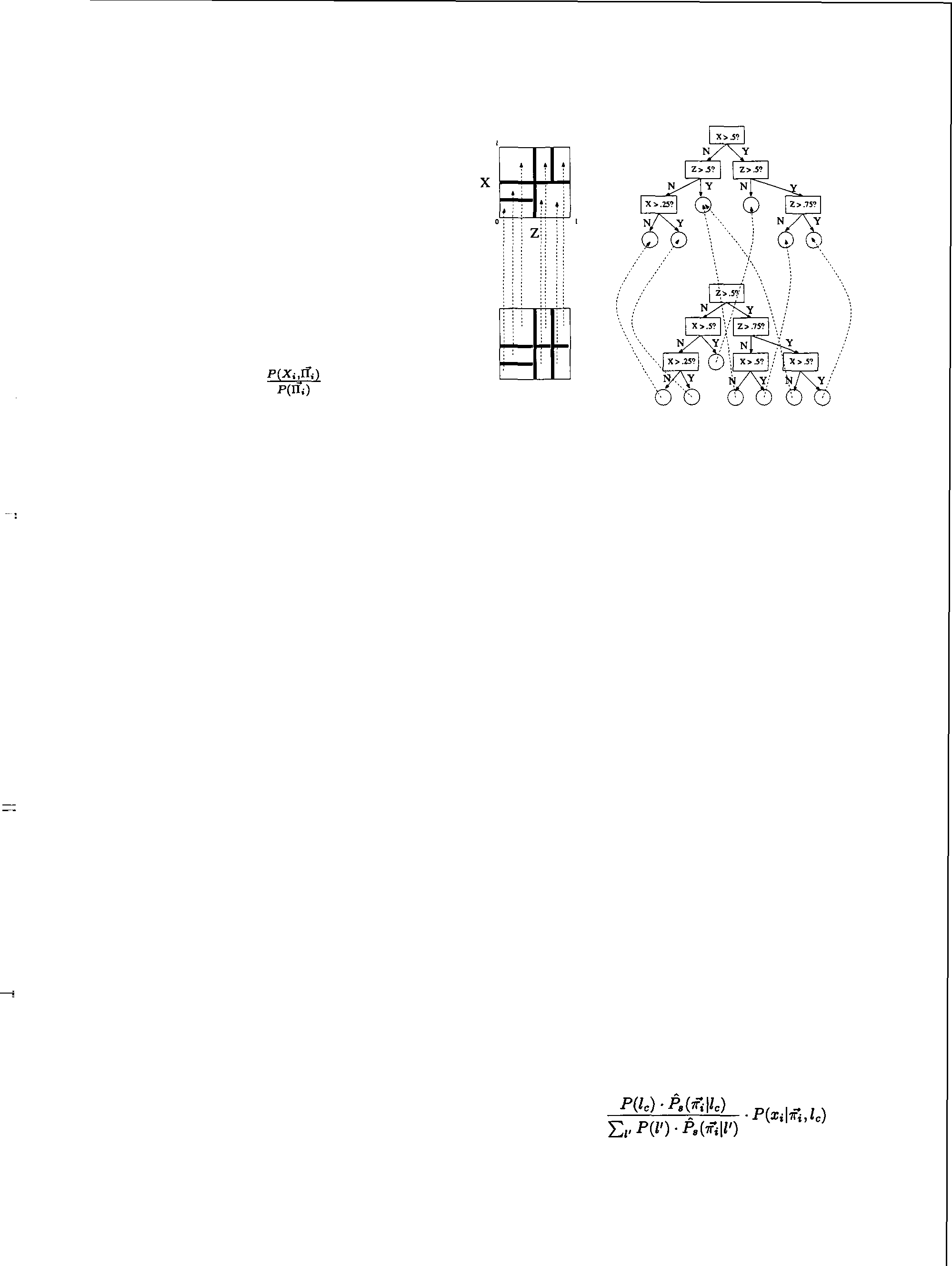
sistent with 1ri the datapoint probably came from.

Somewhat surprisingly, our experimental results show that learning joint density trees and then using them conditionally in this manner frequently leads to more accurate conditional density estimation than the more direct approach of learning and using stratified condi­ tional density trees. One possible explanation of this phenomenon is discussed briefly at the end of the next section.

* 1. Approximately conditionalized joint density trees

The joint density trees discussed in the previous sec­ tion can be learned quickly and they appear to be at least as accurate as stratified conditional density trees in our experiments. However, they are computation­ ally expensive to use, since evaluating the denominator

UAI2002 DAVIES & MOORE 123



Lv P(l') · P(iiil') requires traversing the tree finding all leaves consistent with the known value of 1rj.

If the class of density functions used in the leaves is closed under addition and scalar multiplication, then we can take a density tree modeling P(X;, IT;) and precompute a *marginalized* density tree P(IT;). Such a marginalization algorithm for density trees with constant-density leaves has been used in previous work by Kozlov and Koller on message-passing algo­ rithms for inference in continuous-variable graphical models (Kozlov and Koller, 1997). Once this tree is computed, we can compute the conditional distribu-

tion simply as P(X;I�II;) = , where comput·mg

the numerator and computing the denominator each require locating and evaluating only one leaf distri­ bution in the appropriate tree. Unfortunately, many useful leaf density estimators are not closed under ad­ dition, including those that have factored nonuniform distributions over multiple variables. Marginalizing trees with such leaves results in a marginalized tree whose leaves contain mixture distributions with many components, and evaluating these leaves *can* take a sig­ nificant amount of computational time. Furthermore, for some operations we might wish to perform with density trees, such as sampling or compression, being able to compute P(X;IIT;) as a quotient of two black­ box functions is not particularly helpful. Such oper­ ations are much more naturally performed in terms of leaf probabilities P(LIIT;) and leaf-dependent con­ ditional probabilities P(X;IL,IT;). (For example, sup­ pose we have an algorithm capable of generating a ran­ dom sample from a Gaussian distribution. It is simple to use this routine to generate a random sample from a mixture of Gaussians -first, we randomly choose the mixture component, and then we generate a random sample from the corresponding Gaussian distribution. On the other hand, it is not so straightforward to use it to generate a random sample from a distribution represented as a quotient of two Gaussian mixtures.)

However, in such situations we can still speed up the evaluation of conditional probabilities by creating an auxiliary tree in which each leaf contains a pointer to a single leaf of the original density tree. This auxiliary tree has the same structure as a stratified conditional density tree in that all branching on the parent vari­ ables is performed first, after which all branching is on the child variable. We create the auxiliary tree's structure by first using a marginalization algorithm similar to that employed by Kozlov and Koller. This marginalization algorithm produces a tree in which al branches over X; have been removed, and which con­ tains one leaf for every distinct possible combination of leaves in the original tree that *can* be consistent

with any single 'lfj. We then recursively refine each

Figure 3: An example of a conditionalized joint density tree. Each leaf of the auxiliary tree (bottom) contains a pointer back up to a single leaf in the original density tree (top). The geometrical representation of each tree is shown to the left, and the tree-based representation to the right.

leaf of the resulting marginalized tree by branching on X; until each of the resulting leaves has a nonzero intersection with precisely one of the leaves in the orig­ inal joint density tree. See Figure 3 for an example; see the full paper (Davies, 2002) for further detail. We call the combination of the original joint density tree and the auxiliary tree a *conditionalized joint density tree.*

The auxiliary tree can help speed up the evaluation of conditional probabilities simply by providing (in a sin­ gle, relatively small subtree) pointers to all the leaves in the original joint density tree that are consistent with any given value of IT;. This speedup is roughly similar to that which would be achieved by using a marginalized density tree with mixture models in the leaves to compute P(IT;) -roughly a factor of two or so in our experiments (not shown here). However, we can speed up the conditional evaluation of joint den­ sity trees further by introducing a simple approxima­ tion. Within the context of any subtree *t,* where the branching on X; begins, we *can* approximate the con­ ditional distribution P(IT;il') over each original density tree leaf l' *as* .P,(IT;Il'), the mean of P(IT;il') over all the datapoints consistent with ta's constraints. The conditional density *can* then be be computed approx­ imately as:

# P(x;iii) *�*

= a�P(x;iii,lc)

where lc is the single leaf consistent with both x; and 7fi

*124* DAVIES & MOORE UAI2002



## (as before) and a� is a constant. (An alternative would be to compute a� directly as the average of P(lcl'li'i) over the datapoints consistent with t.'s constraints;

however, this appears to not work quite as well empir­ ically.) When we use a conditionalized joint density tree in this manner, we refer to it as an *approximately conditionalized joint density tree.*

## If each leaf of the original joint tree employs a nonuni­ form distribution over the parent variables, then ob­ taining the conditional distribution P(X;I7ri) from a joint tree using the relationship

P(x;l1ri) = L: P(llii) · P(x;lii, *l)*

*I*

## can acttually result in more accurate conditional den­ sity estimation than possible with stratified condi­ tional density trees, even though the joint density trees are optimized for joint probabilities rather than con­ ditional probabilities. Intuitively, by combining the distributions learned in different leaves using this re­ lationship, we have essentially created a *"soft branch"*

over IT; that helps us to more accurately predictt X; as a function of IT; without acttually splitting the dataset further into completely disjoint subsets. In fact, as our

experimental results will show, conditionalized joint density trees can actually provide more accurate es­ timates than stratified conditional density trees even when the conditionalized joint density trees are used approximately, i.e., even when the "soft branching"

coefficients a� are fixed as constants.

* 1. Leaf types

## In all of our experiments, each leaf represents the dis­ tribution of each discrete variable as a multinomial dis­ tribution independent of all other variables. However, *we* have experimented with a wide variety of distri­ butions with which to represent the densities of con­ tinuous variables within each density tree leaf: con­ stant (i.e. uniform) densities; Gaussians with diago­ nal covariance matrices or general covariance matrices (renormalized so our conditional distributions always integrate to one); exponential distributions; and linear and multilinear distributions. Space restricttions pre­ vent us from discussing all of these possibilities here; see (Davies, 2002) for further detail. Of these, linear and multilinear interpolation appear empirically to be the best density approximators for use in the leaves of stratified conditional density trees and conditional­ ized joint density trees, with multilinear interpolation being slightly more accurate than linear interpolation but also more computationally expensive. With lin­ ear interpolation, each continuous variable is modeled independently, and its density varies linearly within each leaf. With multilinear interpolation, the *d* con-

tinuous variables are modeled jointly by interpolating between 2d densities associated with the corners of the

leaf's bounding hyperbox. In both cases, each distri­ bution to fit is expressed as a mixtture model and then fit with the EM algorithm (Dempster et a!., 1977) to maximize the log-likelihood of the training data. Be­ cause the distribution of each mixture component is fixed and only the prior probabilities of the mixture components are adjusted, EM can be performed rela­ tively quickly - and the log-likelihood is convex, so there are no suboptimal local maxima for EM to get trapped in. In order to keep leaf-learning reasonably fast at the higher levels of the tree where many dat­ apoints lie in each candidate leaf, we resttrict the EM

algorithm to using at most 25 \* 2d datapoints to fit

any d-dimensional multilinear disttribution, or at most 25 \* 2 \* d datapoints to fit any d-dimensional inde­ pendent linear interpolation. Furthermore, we restrict EM to run for at most 10 iterations. Experiments not described in this paper have shown that this subsam­ pling and this limitation on the number of iterations have a negligible effect on the accuracy of the resulting density estimator.

* 1. Smoothin

## The tree-learning algorithms *we* employ are generally oriented towards maximizing the log-likelihood of the data- either just of X; in the case of CART-like and

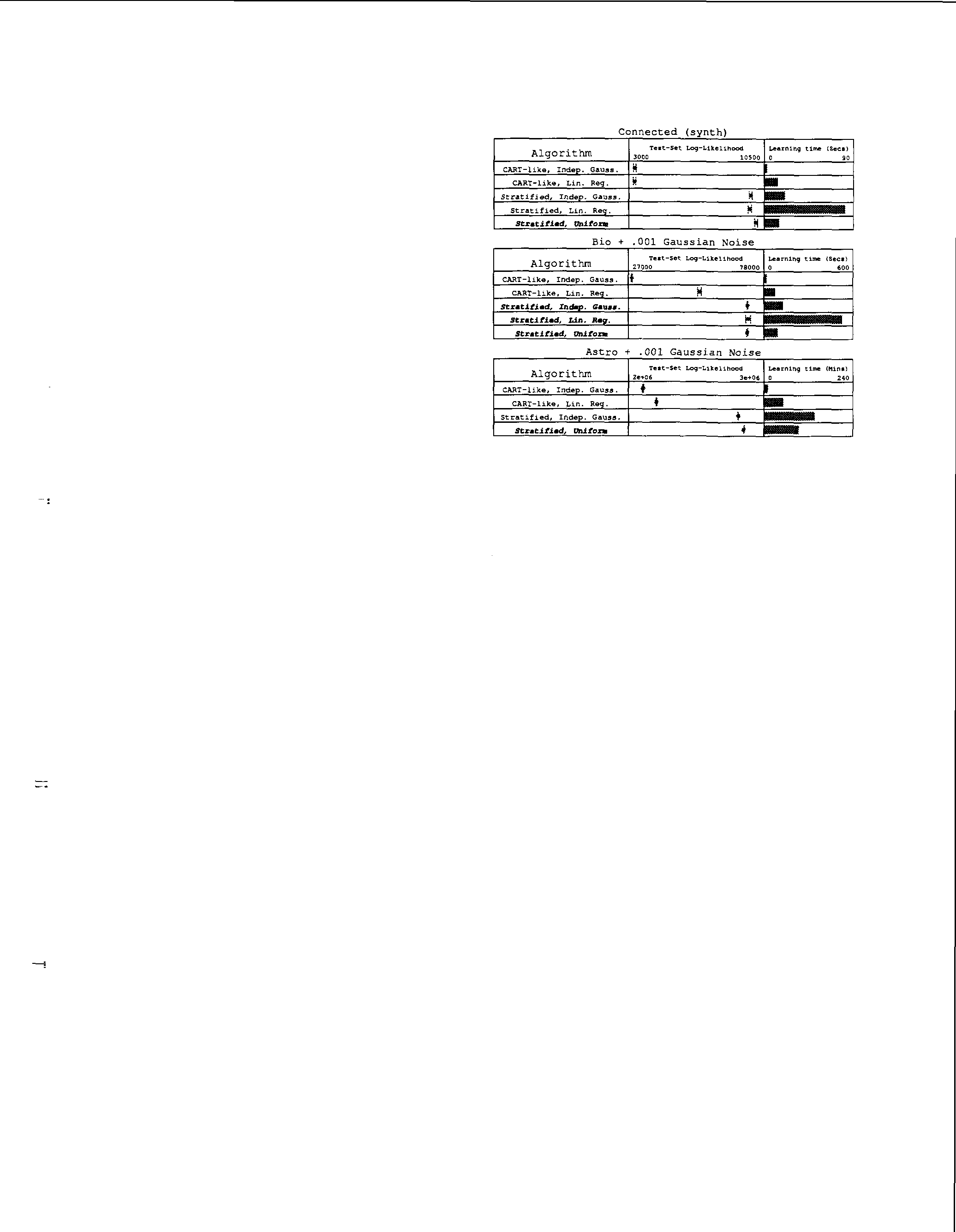
stratified conditional density trees, or of {X;} *U* IT; in the case of joint density trees. If we are using test­

set log-likelihood as our criterion for density estimator quality, such maximum-likelihood estimates can per­ form arbitrarily poorly. Rather than attempt a com­ plex fully Bayesian solution to the problem, we rely on a commonly used and simpler technique for work­ ing around it: namely, we adjust the overall distri­ bution slightly towards the uniform distribution in an ad-hoc fashion. For simplicity, *we* assume some bound­ ing box is known *a priori* for the continuous variables. See (Davies, 2002) for details and a discussion of how to handle other scenarios.

* 1. Experimental results

## In this section we compare the accuracy of the four tree types described above on a simple synthetic dataset and on two large scientific datasets. The "Connected" synthetic dataset was generated by sampling 80,000 datapoints from a mixtture of Gaussians in two di­ mensions. The "Bio" dataset contains data from a high-throughput biological cell assay. There are 12,671 records and 31 variables. 26 of the variables are con­ tinuous; the other five are discrete. Each discrete vari­ able can take on either two or three different possible values. The "Astro" dataset contains data taken from

UA12002 DAVIES & MOORE 125



# the Sloan Digital Sky Survey, an extensive astronomi­ cal survey currently in progress. This dataset contains 111,456 records and 68 variables. 65 of the variables are continuous; the other three are discrete, with ari­ ties ranging from three to 81. See the full version of the paper (Davies, 2002) for further experiments on addi­ tional synthetic datasets, on other modifications of the scientific datasets, and with many other variations of the learning algorithms.

Two minor adjustments were made to each of the sci­ entific datasets before handing them to any of our learning algorithms. First, all continuous variables

were scaled so that all values lie within [0, 1]. This helps put the log-likelihoods we report in context, and

possibly helps prevent problems with limited machine floating-point representation. Second, the value of each continuous value in the dataset were randomly perturbed by adding noise to it - either uniform noise with a range of .001, or Gaussian noise with a standard deviation of .001. This noise was added to eliminate any deterministic relationships or delta functions in the data. The log-likelihood of a continuous dataset exhibiting even a single deterministic relationship be­ tween two variables is infinite when given the correct model; in such a situation, it is not clear how mean­ ingful log-likelihood comparisons between competing learning algorithms would be. Adding two different kinds of noise also allows us to check how sensitive the algorithms' relative performances are to variations in the small-scale details of the datasets.

Figure 4 shows a sample of our experimental results for CART-like vs. stratified conditional density trees. Two different kinds of leaf types are shown for the CART-like trees: Gaussians with constant means, and Gaussians in which the mean is a linear function of the parent variables as determined by linear regres­ sion. For stratified conditional density trees we show results for uniform-density leaves in addition to these two previous leaf types. A 10-fold cross-validation is performed; we show the mean of the log-likelihoods of the test sets, as well as its empirically estimated 95% confidence interval. The best algorithm for a given dataset is shown in bold italics, as well as all others that are not worse than it with at least 95% confi­ dence according to a Student's t-test. In the case of the synthetic "Connected" dataset, the task is sim­ ply to model the conditional distribution of one vari­ able given the other; in the case of the two scientific datasets, the task is to model the joint distribution over all the variables using a Bayesian network with a fixed structure. (These structures had been learned automatically in previous work (Davies and Moore, 2000)). The results show that stratified conditional density trees model the distributions much more accu-

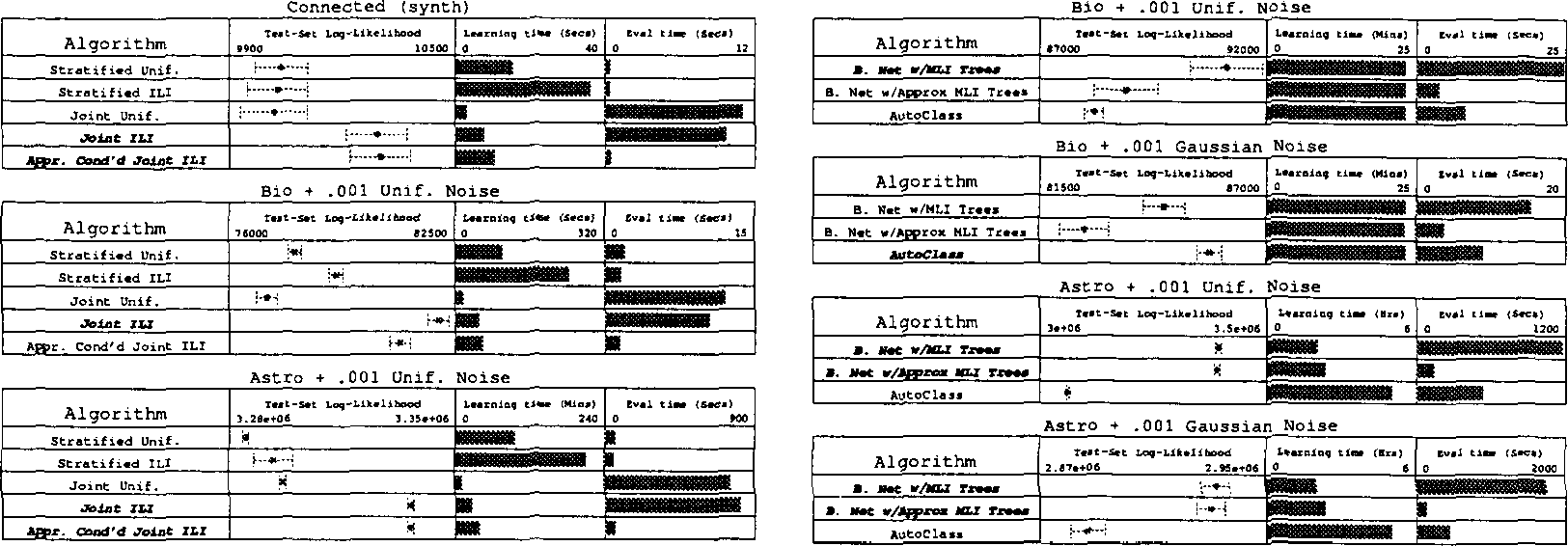
Figure 4: Accuracies and learning times for CART-like vs. stratified density trees.

rately. This is unsurprising in the case of the syn­ thetic dataset, which was generated to have multi­ modal conditional distributions; however, it is inter­ esting to note that the scientific datasets also contain complex conditional distributions not adequately cap­ tured by CART-like conditional density trees. Unfor­ tunately, stratified conditional density trees are also much more computationally expensive to learn; in the case of the Astro dataset, the experiment for stratified conditional density trees employing linear regression in the leaves was aborted because it would have taken several CPU-days to complete. (We omit the results for the scientific datasets with uniform noise added rather than Gaussian; however, they are qualitatively similar.)

Figure 5 shows some of our experimental results com­ paring stratified conditional density trees vs. various forms of joint density trees. Two different leaf distribu­ tions are shown for each tree type: uniform, and inde­ pendent linear interpolations for each variable ("ILl"). These results illustrate several important trends:

* + - Using interpolating distributions within the leaves improves accuracy over uniform distributions.
    - Joint density trees with interpolating leaves are more accurate than stratified conditional density trees, and are fast to learn. Unfortunately, they're much more expensive to evaluate.
    - Approximately conditionalized joint density trees are *still* more accurate than stratified conditional density trees, but are much faster to learn and about as fast to evaluate. Thus, these approxi-

126 DAVIES & MOORE UAJ 2002



## Figure 5: Stratified vs. Joint vs. Approximately Con­ ditionalized Joint density trees.

mately conditionalized joint density trees combine the best features of the three other tree types.

Similar results are obtained for other synthetic datasets and for the scientific datasets with Gaussian noise added (Davies, 2002).

* 1. Learnin Bayesian Network Structures with Interpolatin Conditional Density Trees

## We have developed an iterative Bayesian network structure-learning algorithm capable of using different kinds of density trees for three different phases of the learning task. This algorithm is somewhat similar in spirit to the Sparse Candidate algorithm for learning network structures over discrete variables (Friedman et al., 1999), and can be seen as a heuristic approxima­ tion of steepest-ascent hill-climbing in order to make it computationally feasible. Up to three different kinds of density trees may be used for three different parts of the algorithm:

* + - Fast-to-learn but relatively inaccurate trees can be used to occasionally recompute "steepness" es­ timates in network structure search space, i.e., which arc additions and removals seem promising.
    - Medium-quality trees can be used to compare a new candidate network structure with the best previously found network structure.
    - Expensive, high-quality density trees can be used for the final network parameterization after a promising network structure has been settled upon.

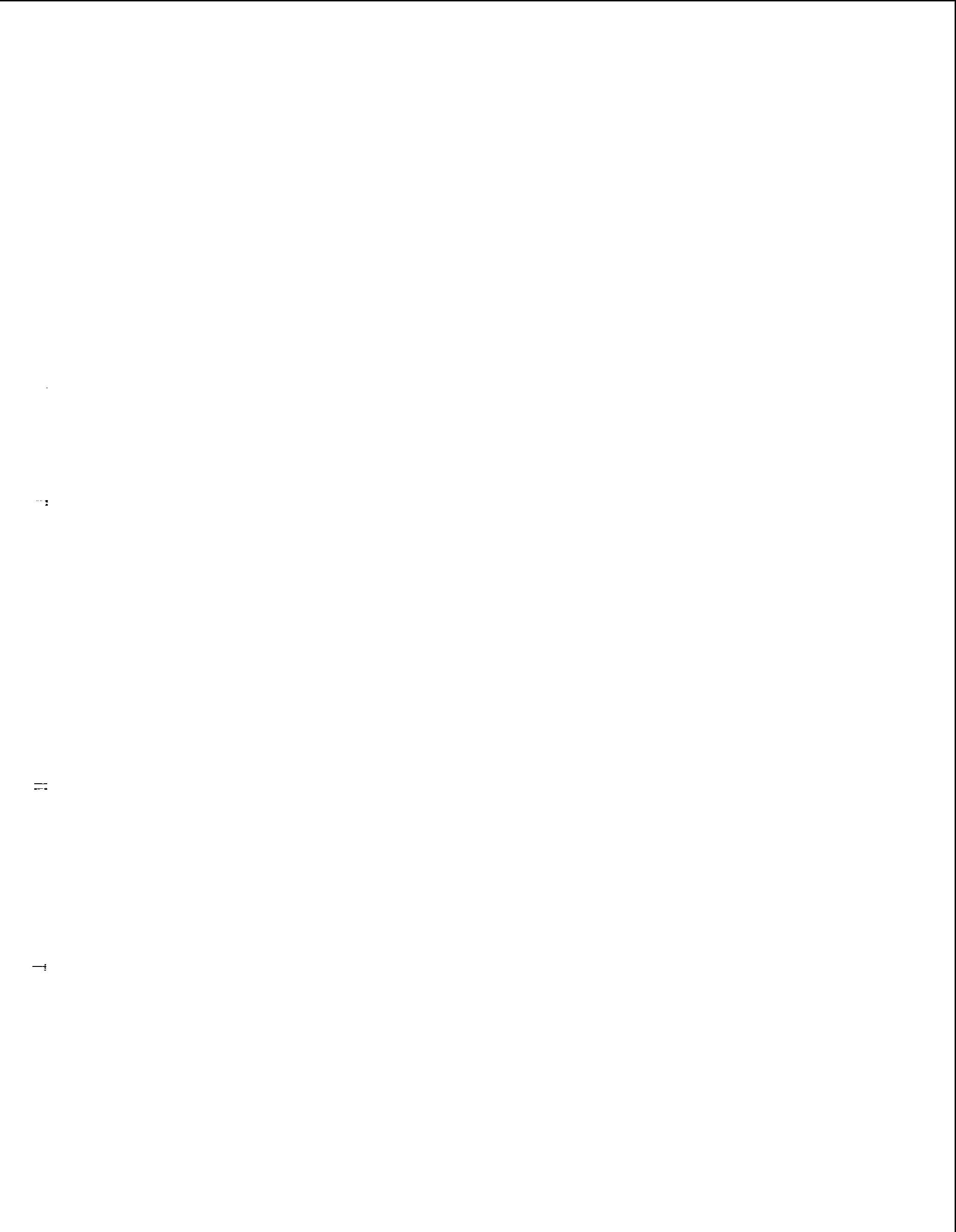
Figure 6: Automatically learned Bayesian networks with density trees vs. global mixture models learned by AutoClass.

Space restrictions preclude a detailed description of this network structure-learning algorithm; see (Davies, 2002) for details. Using this flexible network search algorithm allows us to learn Bayesian networks mod­ eling joint probability distributions over many contin­ uous and discrete variables in a reasonable amount of time. We compare the accuracy of these Bayesian net­ works with that of global mixture models learned over all variables simultaneously by AutoClass (Cheeseman and Stutz, 1996). The results in Figure 6 show that on the higher-dimensional "Astra" scientific dataset, our Bayesian networks provide significantly more accu­ rate density estimation than the global mixture mod­ els, and can be learned and evaluated more quickly as well - even when Gaussian noise is added to the data, which would favor AutoClass's Gaussian mixture mod­ els. The difference on the Astro dataset is even more dramatic when the added noise is uniform. Which of the two approaches works better on the Bio dataset depends on the type of noise addeq; our networks fare better when the added noise is uniform, but AutoClass fares better when the noise is Gaussian.

3 CONCLUSIONS

We have explored a wide variety of tree-based rep­ resentations for conditional density estimation, and shown that they can be used to feasibly learn Bayesian networks over dozens of continuous variables from many thousands of datapoints. In some cases, the resulting models are simultaneously more accu­ rate, faster to learn, and faster to evaluate than global mixture models. We have not yet experi­ mentally compared this approach to previously devel­ oped global discretization-based approaches to learn-

UA12002 DAVIES & MOORE 127



# ing Bayesian networks (e.g. (Friedman and Gold­ szmidt, 1996a), (Monti and Cooper, 1998a)); while we have presented an interesting possible alternative, fur­ ther experimentation is warranted. Numerous other lines of further research are possible; for example, ex­ plicit accuracy/computation tradeoffs *can* be explored for approximately conditionalized joint density trees. See (Davies, 2002) for further discussion of these and other issues.

References

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression 7rees.* Chapman & Hall, 1984.

# Cheeseman and J. Stutz. Bayesian classifica­ tion (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and

* 1. Uthur*asam*y, editors, *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

# Davies. Fast Factored Density Estimation and Com­ pression with Bayesian Networks. Ph.D. Thesis, Carnegie Mellon University, 2002.

S. Davies and A. Moore. Mix-nets: Factored Mixtures of Gaussians in Bayesian Networks with Mixed Con­ tinuous and Discrete Variables. In *Proceedings of the Sixteenth Conference on Uncertainty in Artifi­ cial Intelligence (UAI£000},* 2000.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Max­ imum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society,* B 39:1-39, 1977.

E. Driver and D. Morrell. Implementation of Conti­ nous Bayesian Networks Using Sums of Weighted Gaussians. In *Proceedings of the Eleventh Con­ ference on Uncertainty in Artificial Intelligence (UAI95},* 1995.

N. Friedman and M. Goldszmidt. Discretizing Contin­ uous Attributes While Learning Bayesian Networks. In *Proceedings of the Thirteenth International Con­ ference on Machine Learning,* pages 157-165, 1996a.

N. Friedman and M. Goldszmidt. Learning Bayesian Networks with Local Structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI96},* 1996b.

N. Friedman and I. Nachman. Gaussian Process Net­ works. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI£000},* 2000.

N. Friedman, I. Nachman, and D. Peer. Learn­ ing Bayesian Network Structures from Massive Datasets: The Sparse Candidate Algorithm. In *Pro­ ceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI99},* pages 206-215,

# 1999.

D. Heckerman and D. Geiger. Learning Bayesian net­ works: a unification for discrete and Gaussian do­ mains. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI95},* 1995.

# David Heckerman and Christopher Meek. Embed­ ded Bayesian network clasifiers. Technical Re­ port MSR-TR-97-06, Microsoft Research, Redmond, WA, March 1997.

R. Hoftann and V. Tresp. Discovering Structure in Continuous Variables Using Bayesian Networks. In

D. S. Touretzsky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8.* MIT Press, 1995.

R. Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In *Proceedings of the Second International Conference on Knowl­ edge Discovery and Data Mining (KDD-96},* 1996.

A. Kozlov and D. Koller. Nonuniform dynamic dis­ cretization in hybrid networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI97},* 1997.

S. Monti and G. F. Cooper. A Multivariate Dis­ cretization Method for Learning Bayesian Networks from Mixed Data. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI98},* 1998a.

S. Monti and G. F. Cooper. Learning Hybrid Bayesian Networks from Data. In M. I. Jordan, editor, *Learn­ ing in Graphical Models.* Kluwer Academic P ublish­ ers, 1998b.

S. Monti and G. F. Cooper. A Latent Variable Model for Multivariate Discretization. In *Proceedings of the Seventh International Workship on AI* & *Statistics (Uncertainty 99},* 1999.

J. R. Quinlan. Induction of decision trees. *Machine Learning,* 1:81-106, 1986.