A tree augmented classifier based on Extreme Imprecise Dirichlet Model

G. Corani[a], C. P. de Campos[a]

[a] *IDSIA Manno, Switzerland*
{*giorgio,cassio*}*@idsia.ch*

Abstract

We present TANC, a TAN classifier (tree-augmented naive) based on imprecise probabilities. TANC models prior near-ignorance via the Extreme Imprecise Dirichlet Model (EDM). A first contribution of this paper is the experimental comparison between EDM and the global Imprecise Dirichlet Model using the naive credal classifier (NCC), with the aim of showing that EDM is a sensible approximation of the global IDM. TANC is able to deal with missing data in a conservative manner by considering all possible completions (without assuming them to be missing-at-random), but avoiding an exponential increase of the computational time. By experiments on real data sets, we show that TANC is more reliable than the Bayesian TAN and that it provides better performance compared to previous TANs based on imprecise probabilities. Yet, TANC is sometimes outperformed by NCC because the learned TAN structures are too complex; this calls for novel algorithms for learning the TAN structures, better suited for an imprecise probability classifier.

1. Introduction

Classification is the problem of predicting the *class* of a given object, on the basis of some attributes (*features*) of it. A classical example is the iris problem by Fisher: the goal is to correctly predict the *class*, i.e., the species of Iris on the basis of four features (the length and the width of sepal and petal). In the Bayesian framework, classification is accomplished by updating a prior density (representing the beliefs before analyzing the data) with the likelihood (modeling the evidence coming from the data), in order to compute a posterior density over the classes, which is then used to select the most probable class.

The naive Bayes classifier (NBC) [1] is based on the "naive" assumption of stochastic independence of the features given the class; since the real data generation mechanism generally does *not* satisfy such condition, this introduces a severe bias in the probabilities estimated by NBC. Yet, at least under the 0-1 loss, NBC performs surprisingly well [1, 2]. Reasons for this phenomenon have been provided, among others, by Friedman [3], who proposed an approach to decompose the misclassification error into bias error and variance error; the bias error represents how closely the classifier approximates the target function, while the variance error reflects the sensitivity

of the parameters of the classifier to the training sample. Low bias and low variance are two conflicting objectives; for instance, NBC has high bias (because of the unrealistic independence assumption) but low variance, since it requires to estimate only a few parameters. The point was clearly made also by Domingos and Pazzani [1] who commented about NBC and C4.5 (a classifier with lower bias but higher variance than NBC's): "*A classifier with high bias and low variance will tend to produce lower zero-one loss than one with low bias and high variance, because only the variance's effect will be felt. In this way, the naive Bayesian classifier can often be a more accurate classifier than C4.5, even if in the infinite-sample limit the latter would provide a better approximation. This effect should be especially visible at smaller sample sizes, since variance decreases with sample size. Indeed, Kohavi (1996) has observed that the Bayesian classifier tends to outperform C4.5 on smaller data sets, and conversely for larger ones.*"

Therefore, NBC can be accurate on small and medium data sets, but is then generally outperformed by more complex (i.e, less biased) classifiers on large data sets. A way to reduce the NBC bias is to relax the independence assumption using a more complex graph, like TAN (tree-augmented naive Bayes) [4]. In particular, TAN can be seen as a Bayesian network where each feature has the class as parent, and possibly also a feature as second parent. In fact, TAN is a compromise between general Bayesian networks, whose structure is learned without constraints and NBC, whose structure is determined in advance to be naive (i.e., each feature has the class as the only parent). TAN has been shown to outperform both general Bayesian networks and naive Bayes [4, 5]. However the advantage of TAN over NBC is especially important on medium and large data sets, as predicted by the bias-variance analysis.

In this paper we develop a *credal* version of TAN; the main characteristic of credal classifiers is that they return more classes when faced with a *prior-dependent* instance, i.e., when the most probable class of the instance varies with the prior adopted to induce the classifier. Credal classifiers face prior-dependent instances by returning a set of classes instead of a fragile single class, thus preserving reliability. They are based on a set of priors rather than on a single prior, which removes the arbitrariness involved in the choice of any single prior. The set of priors is modeled using the Imprecise Dirichlet Model (IDM) [6]. The IDM satisfies a number of properties which are desirable to model prior ignorance [7][1].

Two IDM variants have been adopted in credal classifiers: the *global* and the *local* one. The global IDM allows to compute narrower intervals for upper and lower probabilities, but poses challenging computational problems. In fact, tractable algorithms for exactly computing upper and lower probabilities with the global IDM exist for the naive credal classifier [8], but not for general credal networks. On the other hand, the local IDM returns probability intervals which can be unnecessarily wide, but can be easily computed for any network structure.

Recently, the EDM (Extreme Dirichlet Model) [9] has been introduced, which restricts the global IDM to the extreme distributions. The intervals returned by the EDM

[1]More precisely, *near-ignorance*; full ignorance is not compatible with learning, as shown in Section 7.3.7 of Walley [6].

are included (*inner approximation*) in the intervals returned by the global IDM. Interestingly, the EDM enables an easier computation of upper and lower probabilities, compared to the global IDM. Yet, the EDM has not been tested in classification; a first contribution of this paper is a thorough test of the EDM, carried out using the NCC: in particular, we have compared the "original" NCC of [8], based on the global IDM against NCC based on the EDM (NCC-EDM). The results show that NCC and NCC-EDM identically classify the large majority of instances, thus supporting the introduction of the EDM in credal classifiers as sensible and computationally tractable approximation of the global IDM.

However, besides prior ignorance, there is another kind of ignorance involved in the process of learning from data: ignorance about the missingness process. Usually, classifiers ignore missing data; this entails the idea that the missingness process (MP) is non-selective in producing missing data, i.e., it is MAR (*missing at random* [10]). However, assuming MAR cannot be regarded as an objective approach if one is ignorant about the MP. By the term nonMAR we indicate not only that we cannot assume MAR, but more generally that we have no information about the MP. According to the Conservative Updating Rule [11, 12], in order to deal conservatively with nonMAR missing data, it is necessary to consider all the possible replacements for missing data. The latest version of the naive credal classifier [13] does so.

In this paper we present TANC, a credal TAN which (a) models prior ignorance via the EDM and (b) treats missing data as nonMAR, by therefore considering all possible replacements. However, while the number of possible completions increase exponentially with the total number of missing data, the computational complexity of TANC remains affordable thanks to optimized algorithms. For the moment, TANC efficiently deals with nonMAR missing data in the training set; missing data in the test instance need to be completed in all possible ways, and thus the time increases exponentially in the number of such missing data. We leave for future work the development of an algorithm to deal efficiently with missing data in the test instance.

A credal TAN was already proposed in [14]; we refer to that algorithm as TANC*. TANC* was based on the local IDM (probably because of the difficulties to compute the global IDM) and returned a considerably large number of indeterminate classifications [14]. Moreover, TANC* did not deal with nonMAR missing data.

We thoroughly evaluate TANC by experiments on 45 data sets; we compare TANC against the Bayesian TAN, showing that the accuracy of TAN sharply drops on the instances which are indeterminately classified by TANC. Then, we compare TANC with TANC*, via some metric introduced in [15] to compare credal classifiers. In fact, TANC outperforms TANC*; in particular, it is less indeterminate than TANC*, discovering some instances which can be safely classified with a single class and which were classified indeterminately by TANC*. Then, we compare TANC with NCC. It turns out that TANC is outperformed by NCC on several data sets; the reason is that the TAN structure (learned using a traditional MDL algorithm) is sometimes too complex, causing TANC to become excessively indeterminate. We think that a novel algorithm for discovering the TAN structure may significantly improve the TANC performance by designing parameter-parsimonious structures; it could for instance use imprecise probabilities to cautiously decide whether to assign the second parent to a feature or not. However, there are also a few data sets where TANC does outperform NCC;

they contain correlated variables and many instances, as predicted by the bias-variance analysis. Eventually, we present some preliminary results with nonMAR missing data; the performance of TANC in these cases is quite close to that of NCC (the only other classifier able to deal with nonMAR missing data).

2. Notation

A *credal network* is characterized by (a) a directed acyclic graph $\mathcal{G}$, whose nodes are associated to a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ and by (b) a set $\mathcal{K}$ of multinomial distributions so that each $p \in \mathcal{K}$ factorizes as $p(\mathcal{X}) = \prod_i p(X_i|\Pi_i)$, where Π_i denotes the parents of X_i (the factorization can be read as *every variable is conditionally independent of its non-descendants given its parents*).[2]

In the particular case of classification using a naive or a TAN structure, the *class* variable C is the only *root* of the network, i.e., the only node with no parents; there are then several feature variables $\mathcal{Y} = \mathcal{X} \setminus C$. The state space of each variable $X \in \mathcal{X}$ is denoted Ω_X, while a state space for a subset $\mathbf{X} \subseteq \mathcal{X}$ is the Cartesian product $\Omega_{\mathbf{X}} = \prod_{X \in \mathbf{X}} \Omega_X$. For instance, the state space of the class is denoted as Ω_C and the state space for all the features is $\Omega_{\mathcal{Y}}$. Assignments are specified by lowercase letters, such as $x_i \in \Omega_{X_i}$, $\pi_i \in \Omega_{\Pi_i}$ (an assignment to all parents of X_i) or $\mathbf{y} \in \Omega_{\mathcal{Y}}$ (an assignment to all features). An assignment $\mathbf{y}$ with a set of variables as subscript, such as $\mathbf{y}_{\mathbf{X}}$, denotes the projection (or restriction) of that assignment to the variables in the subscript set $\mathbf{X} \subseteq \mathcal{X}$, that is, $\mathbf{y}_{\mathbf{X}} \in \Omega_{\mathbf{X}}$. We further denote by Λ_i the set of children of X_i, and by $\sigma(i)$ all the descendants of X_i (not including itself).

The training data set D contains n instances $\mathbf{x} \in \Omega_{\mathcal{X}}$. We denote by $\mathbf{d}^{\mathbf{z}} = \{\mathbf{x} \in D, \mathbf{z} \in \Omega_{\mathbf{Y}} : \mathbf{z} = \mathbf{x}_{\mathbf{Y}}\}$ the subset of instances of D that have the observations of variables $\mathbf{Y}$ equal to the assignment $\mathbf{z}$. Under this notation, $n_{\mathbf{z}} = |\mathbf{d}^{\mathbf{z}}|$ is the number of instances that are compatible with $\mathbf{Y} = \mathbf{z}$. We allow the training data set to contain missing values, that is, for each instance $\mathbf{x}$ some of its elements may be absent. However, we assume the class label to be always present. A *completion* of an instance $\mathbf{x}$ is an assignment to the missing values such that $\mathbf{x}$ becomes complete. A completion of the data set is a completion for all its instances. We denote by $\mathbf{d}_{\mathcal{X}}$ a possible realization of the training data set (i.e., the observed values plus a possible realization for missing data, if any). In the same way, $\mathbf{d}_{\mathbf{X}}$, with $\mathbf{X} \subseteq \mathcal{X}$, is a realization of the data set restricted to the variables $\mathbf{X} \subseteq \mathcal{X}$.

3. Credal Classification

Learning in the Bayesian framework means to update a prior density (representing the beliefs before analyzing the data) with the likelihood (modeling the evidence coming from the data), in order to compute a posterior density, which can then be used to take decisions. In classification, the goal is to compute $p(C|\mathbf{y})$, i.e., the posterior probability of the classes given the values $\mathbf{y}$ of the features in the test instance[3].

[2] The definition of a credal network may vary depending on the independence concept being used.

[3] The probability should be written more precisely as $p(C|D, \mathbf{y})$, since the classifier has been learned on the training set D. Yet, the dependence on D is omitted to keep a lighter notation.

However, especially on small data sets, Bayesian classifiers might return *prior-dependent* classifications, i.e., they might identify a different class as the most probable one, depending on the adopted prior. Yet, the choice of any single prior entails some arbitrariness and such classifications are therefore fragile. Moreover, often one needs to learn entirely from data without modeling domain knowledge; this is often the case in data mining. The problem is usually faced by choosing a uniform prior, in the attempt of being non-informative; yet, it can be argued that the uniform prior models *indifference* rather than ignorance ([6], Sec. 5.5.1). In fact, the uniform prior implies a very precise statement about the equal probability of the different states, which can lead to unsafe conclusions if their effective distribution is far from uniform and the sample size is not large enough to cancel the effect of the prior.

Credal classifiers consider a set of prior densities (*prior credal set*), instead of a unique prior; in this way, they model *prior ignorance*. The prior credal set (usually modeled by the IDM) is then turned into a set of posteriors by element-wise application of Bayes' rule.

Because we deal with sets of densities, a decision criterion must come in place to perform the classification. Under the *maximality* [6] criterion, class c_1 *dominates* class c_2 if $p(c_1)$ is larger than $p(c_2)$ for *all* the densities in the set. More precisely, given the values $\mathbf{y}$ of the features, c_1 dominates c_2 iff: $[\min_{p \in \mathcal{K}}(p(c_1|\mathbf{y}) - p(c_2|\mathbf{y})) > 0]$, where we have denoted by $\mathcal{K}$ the posterior credal set.

Credal classifiers return the classes that are *non-dominated*; for a given instance, there can be one or more non-dominated classes. In the first case, the classification is *determinate*; in the latter, *indeterminate*. In fact, credal classifiers distinguish *hard-to-classify* instances (which are prior-dependent and require more classes to be safely classified), from *easy-to-classify* ones (which can be safely classified with a single class). The set of non-dominated classes is detected by pairwise comparisons, as shown in Figure 1.

IDENTIFICATION OF NON-DOMINATED CLASSES
Input: the feature values $\mathbf{y}$
Output: the non-dominated classes.

1. set NonDominatedClasses := Ω_C;
2. for class $c_1 \in \Omega_C$
   - for class $c_2 \in \Omega_C$, $c_2 \neq c_1$
     - If $\min_{p \in \mathcal{K}}(p(c_1|\mathbf{y}) - p(c_2|\mathbf{y})) > 0$, c_1 dominates c_2, which is dropped from NonDominatedClasses;
3. return NonDominatedClasses.

Figure 1: Identification of non-dominated classes via pairwise comparisons.

4. Variants of the Imprecise Dirichlet Model

Credal classifiers usually adopt the IDM to model the prior credal set. In the following we show the differences between three IDM variants (local, global, and EDM). Let us consider the credal network $C \rightarrow F$; it requires the definition of the credal sets $K(C)$ and $K(F|C)$. We denote as c and f generic states of C and F, respectively. We denote by $\theta_{c,f}$ the unknown chances of the multinomial joint distribution of C and F, by $\theta_{f|c}$ the chance of $F = f$ conditional on c and by θ_c the chance of $C = c$.

Let us consider the computation of the marginal probability $p(c)$ in the precise Bayesian setting. The prior probability $p(\theta_C)$ is a Dirichlet distribution ($\propto \prod_c \theta_c^{\alpha_c - 1}$). A precise value of α_c is specified for each class, respecting the constraints $\forall c: \ \alpha_c > 0$ and $\sum_c \alpha_c = s$, where s can be regarded as the number of *hidden samples* (or hidden instances) and α_c as the proportion of hidden samples having value c. The likelihood is *proportional* to $\prod_c \theta_c^{n_c}$; the posterior, obtained by multiplying prior and likelihood, has the same form of the prior (i.e., it is a Dirichlet density), but with coefficients α_c replaced by $\alpha_c + n_c$. The probability of state c, computed by taking expectation over the posterior density, is:

$$p(c) = \frac{n_c + \alpha_c}{n + s}. \tag{1}$$

Now, we move to imprecise probabilities. Both the local and the global IDM allow each parameter α_c to vary within the interval $(0, s)$, under the constraint $\sum_c \alpha_c = s$. The credal set $K(\theta_c)$ contains therefore *all* the Dirichlet densities which satisfy $\forall c: \alpha_c > 0$ and $\sum_c \alpha_c = s$. Both the local and the global IDM estimate the probability $p(c)$ as ranging inside the interval:

$$p(c) = \left[\frac{n_c}{n + s}, \frac{n_c + s}{n + s} \right], \tag{2}$$

thus defining the credal set $K(C)$. The EDM restricts the possible priors to the extremes of the IDM; it allows each α_c to take only the *extreme* values of 0 or s (always under the constraint $\sum_c \alpha_c = s$), dropping therefore the intermediate distributions. The EDM returns two possible values for $p(c)$: $\frac{n_c}{n+s}$ and $\frac{n_c+s}{n+s}$, i.e., the two extremes of Eq.(2). The EDM assumes in fact that the s hidden instances have the same value of C, and that there is ignorance about *which* value it is. The credal set $K(C)$ built by the EDM contains as many distributions as the number of states c.

Let us now focus on the computation of conditional probabilities. We have to introduce the parameters α_{cf}, which can be regarded as the proportion of hidden instances having state c for C and f for F. The local IDM lets the α_{cf} vary between 0 and s, under the constraints $\forall c: \sum_f \alpha_{cf} = s$. It estimates the conditional probabilities analogously to formula (2):

$$p(f|c) = \left[\frac{n_{cf}}{n_c + s}, \frac{n_{cf} + s}{n_c + s} \right], \tag{3}$$

thus defining the conditional credal set $K(F|C)$. The local IDM produces a local credal set $K(f|c)$ for each c; such credal sets are independent both from each other and from $K(C)$. The network is therefore *locally and separately specified*.

The *global* IDM is based, for each c and f, on a prior credal set for the *joint* chance $\theta_{c,f}$; each prior of the credal set factorizes as $p(\theta_{c,f}) = p(\theta_c)p(\theta_{f|c})$. Yet, given a certain $p(\theta_C)$ (defined by a set of α_c), only certain $p(\theta_{f|c})$ factorize as required, namely those satisfying the constraint $\forall c : \sum_f \alpha_{cf} = \alpha_c$.

For a specific choice of α_c, the global IDM estimates the conditional $p(f|c)$ as:

$$p(f|c) = \left[\frac{n_{cf}}{n_c + \alpha_c}, \frac{n_{cf} + \alpha_c}{n_c + \alpha_c} \right]. \tag{4}$$

Because of the constraints existing between the credal set of marginal and conditional distributions, the network is *not* locally neither separately specified. The global IDM, when applied to a credal network, estimates narrower posterior intervals than the local IDM and leads to less indeterminacy in classification. Yet, the computation of upper and lower *joint* probabilities becomes more difficult, as it cannot be done locally; the NCC is to our knowledge the only case where the computation of upper and lower probabilities is known to be tractable under the global IDM. The local IDM returns wider intervals but enables a much easier computation, because it manages independently the parameters of the different credal sets.

The EDM, which restricts the prior to the extreme distributions of the IDM, allows the coefficients α_{cf} to assume only two values: 0 or α_c, always under the constraint $\forall c : \sum_f \alpha_{cf} = \alpha_c$ inherited from the global IDM. When applied to a single variable, the EDM extremes correspond to the same extremes of the global IDM; however, when applied to a credal network, it returns intervals[4] that are included (or at most equivalent) in the intervals computed by the global IDM [9]. For a credal network, the EDM in fact models the s hidden instances as s *identical* rows of missing data; the ignorance (which generates the credal set) is about the values they contain.

5. IDM vs. EDM: empirical comparison on NCC

The EDM still lacks an experimental validation, as recognized also by [9]. To test the EDM in classification, we have implemented NCC with the EDM (NCC-EDM) and then we have compared it with the traditional NCC, based on the global IDM. We have performed the experiments by reworking the code of the open-source JNCC2 software [16].

NCC-EDM adopts a restricted credal set compared to NCC; therefore, it generally detects a higher minimum when checking credal dominance between c_1 and c_2: $[\min_{p \in \mathcal{K}}(p(c_1|\mathbf{y}) - p(c_2|\mathbf{y})) > 0]$. If the minimum found by NCC is < 0 while the minimum found by NCC-EDM is > 0, NCC-EDM detects credal-dominance and drops c_2 from the non-dominated classes, while NCC retains c_2 as non-dominated. Yet, this does not necessarily affect the final set of non-dominated classes: NCC could later find that c_2 is dominated by a say c_3, and then drop c_2 from the non-dominated classes. However, if this does not happen, the two classifiers return different sets of non-dominated classes.

[4]EDM returns only extremes. In the explanations, we denote the interval of EDM as the interval induced by such extremes.

Data set	Different classifications (%)	Data set	Different classifications (%)
anneal	0.0%	labor	0.0%
audiology	22.6%	letter	0.0%
autos	1.0%	lymphography	0.0%
balance-scale	0.0%	pasture	0.0%
breast-cancer	0.0%	segment	0.0%
credit-rating	0.0%	soybean	1.2%
german_credit	0.0%	squash-stored	0.0%
grub-damage	0.0%	squash-unstored	0.0%
heart-statlog	0.0%	white-clover	0.0%
hepatitis	0.0%	wisc-breast-cancer	0.0%
hung-heart	0.0%	zoo	0.0%
iris	0.0%		

Table 1: Percentage of instances where NCC detects a different set of non-dominated classes, when IDM or EDM is used.

We have compared the classifications issued by NCC and NCC-EDM on 23 data sets from the UCI repository [17][5].; the results are reported in Tab.1. On 22 data sets out of 23, the percentage of credal-dominance tests which receive a different answer from NCC-EDM and NCC is smaller than 1.2%; the percentage of instances over which the two models return a different set of non-dominated classes is about 0.01%. The overall number of credal-dominance tests performed by each classifier is in the order of 10^6, while the total number of classified instances is in the order of 10^5.

However, on the *audiology* data set, NCC and NCC-EDM do return different sets of non-dominated classes in about 23% of the instances. The data set has 226 instances, 69 features and *many* classes (24); several classes are observed only once or twice and moreover most features have *very* skewed distributions (e.g., n_{f_0}=224; n_{f_1}=2). It follows that many conditional frequencies are sharp zeros, thus magnifying the role of the model of prior ignorance. It is reasonable that, under such peculiar conditions, the two models of ignorance lead to different classifications. Still, we can conclude that NCC-EDM is a *close* approximation of NCC.

6. Tree Augmented Naive Credal Classifier

The TAN structure has the characteristic that each feature has at least C as parent and at most one other parent constituted by another feature; this definition actually allows forest of trees (for example, the Naive structure is a subcase of a TAN). TANC is constituted by a credal network over a TAN graph. As described in Section 3, TANC must conduct pairwise comparison to detect credal-dominance; for every comparison between two classes, the dominance test must consider (a) all possible completions of

[5] `http://archive.ics.uci.edu/ml/`

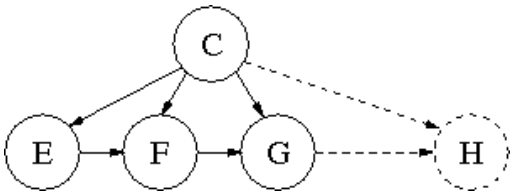

Figure 2: Simple example of TAN structure.

the training data (because missing data of the training set are nonMAR) and (b) all prior densities belonging to the EDM. The credal dominance condition can be rewritten as:

$$\min_{\mathbf{d}_{\mathcal{X}},\boldsymbol{\alpha}} (p(c_1|\mathbf{y}) - p(c_2|\mathbf{y})) > 0, \tag{5}$$

because the distributions $p \in \mathcal{K}$ are completely defined by $\mathbf{d}_{\mathcal{X}}$ and $\boldsymbol{\alpha}$. Using the fact that $p(\mathbf{y})$ is positive and does not affect the sign of the formula, we obtain

$$\min_{\mathbf{d}_{\mathcal{X}},\boldsymbol{\alpha}} (p(\mathbf{y}|c_1)p(c_1) - p(\mathbf{y}|c_2)p(c_2)) > 0. \tag{6}$$

Under the EDM, the parameters $\boldsymbol{\alpha}_C = \{\alpha_{c_1}, \alpha_{c_2}\}$ can only take the two extreme values $\{\alpha_{c_1} = s, \alpha_{c_2} = 0\}$ and $\{\alpha_{c_1} = 0, \alpha_{c_2} = s\}$. We compute Equation (6) for each of these two configurations, which removes any dependency between $p(c_1)$ and $p(c_2)$ (as there are no missing values in the class), obtaining

$$p(c_1) \cdot \left(\min_{\mathbf{d}_{\mathcal{Y}}^{c_1},\boldsymbol{\alpha}_{c_1}} p(\mathbf{y}|c_1) \right) - p(c_2) \cdot \left(\max_{\mathbf{d}_{\mathcal{Y}}^{c_2},\boldsymbol{\alpha}_{c_2}} p(\mathbf{y}|c_2) \right) > 0, \tag{7}$$

which is possible because $p(\mathbf{y}|c_1)$ only depends on the α's related to the class c_1 (which we denote $\boldsymbol{\alpha}_{c_1}$) and on the data of instances with class c_1, while $p(\mathbf{y}|c_2)$ depends on $\boldsymbol{\alpha}_{c_2}$ and counts from instances with class c_2 ($\mathbf{d}_{\mathcal{Y}}^{c_1}$ and $\mathbf{d}_{\mathcal{Y}}^{c_2}$ are obviously disjoint – the values $\alpha_{c_1}, \alpha_{c_2}$ related to C itself are actually fixed because the expression is evaluated for each configuration). The final answer is obtained by taking the minimum of the left-hand of Expression (7) among the two attempts.

We illustrate the execution of the TANC by using the simple example of Figure 2. The example has C as class and E, F, G as features (do not consider the dashed part containing H at first). For ease of expose, we suppose that the data set is complete and we denote as n_{cxz} the number of instances having c, z and x as states for the class and the generic nodes Z and its parent X, respectively. The value of the features in the test instance are $\mathbf{y} = (e, f, g)$. Let $\overline{e}, \overline{f}, \overline{g}$ be respectively the states of E, F, G that are not in $\mathbf{y}$. Given a class state $C = c$, suppose our target is to obtain $\min_{\mathbf{d}_{\mathcal{Y}}^{c},\boldsymbol{\alpha}_c} p(\mathbf{y}|c) = \min_{\boldsymbol{\alpha}_c} p(\mathbf{y}|c)$ (the maximization would be analogous). In the EDM, there are two cases for consideration: $\alpha_c = 0$ and $\alpha_c = s$. We suppose that $\alpha_c = s$, as the computation with $\alpha_c = 0$ is very simple (the minimization vanishes), because the data are complete and the solution would become the frequencies.

It is worth noting that the EDM works in the same way as including a *hidden* instance of weight s where all the variables are missing. $\alpha_c = s$ is equivalent to setting

the class of this hidden instance to c. The parameters $\alpha_{cxz} \in \{0, s\}$ correspond to the EDM counts for X, Z, that is, $\alpha_{cxz} = s$ exactly when the EDM hidden instance is completed with $X = x$ and $Z = z$, and zero otherwise. There are parameters α_{cxz} for every variable Z and parent X and every state z, x. Under the hidden instance analogy for the EDM, the minimization is done over the possible completions of that hidden instance, which induce the values $\boldsymbol{\alpha}_c$ (here $\boldsymbol{\alpha}_c$ means all α's related to class c). Because of the factorization properties of the network, we have:

$$\min_{\boldsymbol{\alpha}_c} p(\mathbf{y}|c) = \min_{\boldsymbol{\alpha}_c} \left(p(e|c) \cdot p(f|e,c) \cdot p(g|f,c)\right) =$$

$$= \min_{\boldsymbol{\alpha}_c} \left(\frac{n_{ce} + \alpha_{ce}}{n_c + s} \cdot \frac{n_{cef} + \alpha_{cef}}{n_{ce} + \alpha_{ce}} \cdot \frac{n_{cfg} + \alpha_{cfg}}{n_{cf} + \alpha_{cf}} \right), \tag{8}$$

subject to the EDM constraints:

$$\sum_z \alpha_{cxz} = \alpha_{cx}, \qquad \sum_x \alpha_{cxz} = \alpha_{cz}, \qquad \sum_{xz} \alpha_{cxz} = s, \qquad \forall xz:\ \alpha_{cxz} \in \{0, s\}.$$

If we were dealing with the maximization instead of the minimization, there is a simple way to solve the optimization of Equation (8): $s = \alpha_{ce} = \alpha_{cef} = \alpha_{cfg}$ achieves the maximum value (to show that this solution is always right, we just apply the following property throughout: $\frac{v_1}{v_2} \leq \frac{v_1+k}{v_2+k}$ if $k > 0$ and $\frac{v_1}{v_2} \leq 1$, which implies that choosing all α's equal to s in Equation (8) is the best option). However, we cannot do a similar straightforward idea for the minimization. For instance, if we try to separately minimize the numerator and maximize the denominator, then we would have to set $\alpha_{cf} = s$, $\alpha_{cef} = 0$ and $\alpha_{cfg} = 0$ (in this example the value assigned to α_{ce} cancels out between the first and second fraction, so it can be set to zero), and this would imply that α_{cg} must be equal to zero, because $\alpha_{c\overline{f}g} \leq \alpha_{c\overline{f}} = 0$ and $\alpha_{cg} = \alpha_{cfg} + \alpha_{c\overline{f}g}$. Therefore, an eventual TANC with the extra node H (the dashed part) would not be able to maximize the denominator of $p(h|g,c) = \frac{n_{cgh}+\alpha_{cgh}}{n_{cg}+\alpha_{cg}}$ by setting $\alpha_{cg} = s$, and thus we cannot separately maximize the denominators while minimizing the numerators (such naive idea only works if there are up to three features, but it does not necessarily work with four features or more).

The previous discussion justifies the need of a specialized algorithm that is able to select how to fill the elements $\boldsymbol{\alpha}_c$ appropriately to minimize the probability of the features given the class. A straightforward approach would take all possible exponential completions of the hidden instance (if we have m features, there would be 2^m possible completions), but fortunately there is a much faster idea that makes use of a decomposition property: if we fix the completion of a feature F, the completions of the children of F can be done independently of the completions of the ancestors. In view of this characteristic, it is possible to devise a bottom-up algorithm over the tree of features that computes the minimization locally to each node by assuming that the parent's missing data are already completed (in fact, the computation is done for each parent completion, like in a dynamic programming idea). The local computation at an intermediate node X_j computes $\phi_{X_j}(\alpha_{cy_j}) = \min_{\boldsymbol{\alpha}_c} p(\mathbf{y}_{\sigma(\mathbf{j})}|y_j, c)$, for every α_{cy_j} ($y_j \in X_j$ is the observed state of X_j in the test instance and $\mathbf{y}_{\sigma(\mathbf{j})}$ are the observed states of all descendants of X_j in the test instance). We first explain the algorithm by

using the same example. We start from the leaf G, where by definition $\phi_G(\alpha_{cg}) = 1$ (because G has no children). Then we process the node F, where the minimization is computed for each completion of F, over all its children (in this example only G) as

$$\phi_F(\alpha_{cf} = s) = \min_{\boldsymbol{\alpha}_c} (p(g|f,c) \cdot \phi_G(\alpha_{cg})) = \min_{\alpha_{cfg}} \left(\frac{n_{cfg} + \alpha_{cfg}}{n_{cf} + s} \cdot 1 \right) = \frac{n_{cfg} + 0}{n_{cf} + s},$$

$$\phi_F(\alpha_{cf} = 0) = \min_{\boldsymbol{\alpha}_c} (p(g|f,c) \cdot \phi_G(\alpha_{cg})) = \frac{n_{cfg} + 0}{n_{cf} + 0} \cdot 1 = \frac{n_{cfg}}{n_{cf}},$$

(note that $\alpha_{cfg} \leq \alpha_{cf}$, so it becomes zero when $\alpha_{cf} = 0$). At this stage, $\phi_F(\alpha_{cf})$ equals to $\min_{\boldsymbol{\alpha}_c} p(g|f,c)$, that is, the probability of the descendants of F (which is just G) given itself and the class. With these two values calculated, we proceed up in the tree to process E, again for each completion:

$$\phi_E(\alpha_{ce} = s) = \min_{\boldsymbol{\alpha}_c} (p(f|e,c) \cdot \phi_F(\alpha_{cf})) = \min_{\alpha_{cef}, \alpha_{cf}} \left(\frac{n_{cef} + \alpha_{cef}}{n_{ce} + s} \cdot \phi_F(\alpha_{cf}) \right),$$

subject to the EDM constraints, and thus the minimization can be tackled by inspecting the possible pairs $(\alpha_{cf}, \alpha_{cef}) \in \{(s,s), (0,0)\}$ (the pair $(0,s)$ is impossible because $\alpha_{cf} \geq \alpha_{cef}$ and the pair $(s,0)$ is impossible because $\alpha_{ce} = s$ and $\alpha_{cef} = 0$ imply $\alpha_{cf} = 0$), and

$$\phi_E(\alpha_{ce} = 0) = \min_{\boldsymbol{\alpha}_c} (p(f|e,c) \cdot \phi_F(\alpha_{cf})) = \frac{n_{cef} + 0}{n_{ce} + 0} \cdot \min_{\alpha_{cf}} \phi_F(\alpha_{cf}),$$

($\alpha_{ce} = 0$ implies that $\alpha_{cef} = 0$) which is done by inspecting $\alpha_{cf} \in \{0, s\}$. Here, $\phi_E(\alpha_{ce})$ equals to $\min_{\boldsymbol{\alpha}_c} p(f,g|e,c)$ (the descendants of E are F and G). The final step over the class obtains the desired result:

$$\phi_C(\cdot) = \min_{\boldsymbol{\alpha}_c} (p(e|c) \cdot \phi_E(\alpha_{ce})) = \min_{\alpha_{ce}} \left(\frac{n_{ce} + \alpha_{ce}}{n_c + s} \cdot \phi_E(\alpha_{ce}) \right),$$

which is done by inspecting $\alpha_{ce} \in \{0, s\}$ and equals to $\min_{\boldsymbol{\alpha}_c} p(e,f,g|c)$ (the descendants of C are all the features). This last step is performed just for the case where $\alpha_c = s$, as it is assumed in this example.

Because we take the Extreme IDM as model for the priors, $\boldsymbol{\alpha}$ only assumes extreme values. As already mentioned, it is possible to tackle the problem by introducing a new instance of weight s to the training set that is completely missing. Because this new hidden instance of missing values has also missing class, it could introduce a dependence between the minimization and the maximization of Equation (7). However, it suffices to solve the optimization for every possible completion of the missing value of the class in this hidden instance (there are just two extremes). Thus we calculate, for every completion of the class in the hidden instance, the equation

$$p(c_1) \cdot \min_{\mathbf{d}_{\mathcal{Y}}^{c_1}} p(\mathbf{y}|c_1) - p(c_2) \cdot \max_{\mathbf{d}_{\mathcal{Y}}^{c_2}} p(\mathbf{y}|c_2). \tag{9}$$

The minimization and the maximization are over every possible completion of the data (including the hidden instance, which now has a known class). Equation (9) differs

ALGORITHM TO COMPUTE THE DOMINANCE TEST
Input: $c_1, c_2, \mathbf{y}$ (assumed to be complete), and the training data without missing classes (the hidden instance is the only with missing class).
Output: the value of the dominance test as in Equation (9).

1. Complete the class of the hidden instance first with c_1, and then with c_2, and do the following for each case:
   (a) Using a reverse topological order of the tree nodes (that is, bottom-up in the tree), for each node X_j (besides the class), do

$$\forall_{\mathbf{d}^{c_1}_{X_j}}\ \phi_j(\mathbf{d}^{c_1}_{X_j}) = \begin{cases} 1, & \text{if } X_j \text{ is a leaf,} \\ \prod_{X_i \in \Lambda_j} \min_{\mathbf{d}^{c_1}_{X_i}} \left(\frac{n_{c_1 y_i y_j}}{n_{c_1 y_j}} \phi_i(\mathbf{d}^{c_1}_{X_i}) \right), & \text{otherwise.} \end{cases}$$

   and equivalently in the maximization case (of course replacing the class to c_2). It is important to keep in mind that $\frac{n_{c_1 y_i y_j}}{n_{c_1 y_j}}$ (as well as the counts of steps 2 and 3), already considers data completions, and thus takes into account the EDM.
   (b) Multiply the values of variables that have only the class as parent:

$$\min_{\mathbf{d}^{c_1}_{\mathbf{y}}} p(\mathbf{y}|c_1) = \phi_C(\cdot) = \prod_{X_i \in \Lambda_C} \min_{\mathbf{d}^{c_1}_{X_i}} \left(\frac{n_{c_1 y_i}}{n_{c_1}} \phi_i(\mathbf{d}^{c_1}_{X_i}) \right),$$

   and similarly for the maximization, that is, $\varphi_C(\cdot) = \max_{\mathbf{d}^{c_2}_{\mathbf{y}}} p(\mathbf{y}|c_2)$ is computed by replacing minimizations to maximizations (and the class from c_1 to c_2).
2. Return the minimum value of $p(c_1)\phi_C(\cdot) - p(c_2)\varphi_C(\cdot)$, where $p(c_2) = \frac{n_{c_2}}{n}$ and $p(c_1) = \frac{n_{c_1}}{n}$, over the two executions of step 1 (for the hidden instance respectively completed with class c_1 and c_2).

Figure 3: Algorithm to compute Equation (9).

from Equation (7) in the sense that there is no $\boldsymbol{\alpha}$ anymore. The EDM is processed by the additional hidden instance, and that is automatically resolved by the possible completions of the data. Using this property, we can use the very same idea to treat nonMAR missing data, as well as the EDM. For that reason, we describe an algorithm to compute Equation (9) instead of Equation (7) and we let the hidden instance and its completions to take care of the EDM. Differently from the example just discussed, the intermediate values of the algorithm (those computed by the functions ϕ) are not described in terms of α's, but in terms of the possible completions of the data, which already accounts for the α's). Apart from that, the algorithm works just as in the previous example. The description of the algorithm is given in Figure 3. Technical details and its correctness are presented in the Appendix A.

We point out that, if the data set is complete, the only missing data that must be processed by the algorithm are those introduced by the hidden instance (for the treatment of the EDM). In such case, the complexity of the method is clearly linear in the

input size, as there is a constant number of computations by variable (there are only two ways of completing the data by variable). In the presence of missing data, the idea spends exponential time in the number of missing data of two linked variables, which is already much better than an overall exponential but still slow for data sets with many missing values. Using extra caches and dynamic programming, it might be possible to further reduce this complexity to exponential in the number of missing values of a single variable.

When a count $n_{cy_iy_j}$ (for the class $C = c$, $X_i = y_i$ and its parent $X_j = y_j$) is zero, there are no observations for estimating the conditional probability $P(y_i|c, y_j)$, which generates a sharp zero during the minimization of (6); therefore, $p(c_1|\mathbf{y})$ in Eq.(6) goes to 0 as well, preventing c_1 to dominate any other class, regardless the information coming from all the remaining features. By adding an artificial epsilon to the counts $n_{cy_iy_j}$, we avoid a single feature to lead the posterior probability of a class to zero. Such a strategy improves the empirical accuracy of both NCC and TANC, although it is more important in the second case, as the TAN structure is more complex and faces zero counts more frequently (for instance, a single zero count for a state of a variable with children is enough to make all the corresponding parameters of the children vacuous, as there are no data to learn them).

7. Experiments

We have performed experiments on 45 UCI data sets, covering a wide spectrum of number of instances (24 - 12960), features (1-69) and classes (2-24). The performance has been measured via 10-fold cross-validation. Since our classifiers (like the standard Bayesian networks) need *discrete* features, we have discretized the numerical features using supervised discretization [18]. We have compared TANC against three competitors: (1) the Bayesian TAN; (2) TANC* (i.e., the TAN based on imprecise probabilities proposed in [14]); (3) NCC. The details are given in Appendix B.

The overall performance of a credal classifier is fully characterized by four indicators, as explained in [13]:

- *determinacy*, i.e., the percentage of instances determinately classified;
- *single-accuracy*, i.e., the accuracy on the *determinately* classified instances;
- *set-accuracy*, i.e., the accuracy on the *indeterminately* classified instances;
- *indeterminate output size*: the average number of classes returned on the indeterminately classified instances.

Note that set-accuracy and indeterminate output size are meaningful only if the data set has more than two classes.

However, how to compare a credal and a precise classifier is still an open problem. Following the approach of [13], we compare TANC and TAN by just evaluating separately the accuracy of TAN on the instances classified determinately and indeterminately by TANC. The rationale is that, if TANC is good at separating hard-to-classify from and easy-to-classify instances, TAN should be less accurate on the instances indeterminately classified by TANC.

Instead, there are two metrics for comparing credal classifiers, which have been proposed in [15]. The first metric, borrowed from multi-label classification[6], is the *discounted-accuracy*:

$$\text{d-acc} = \frac{1}{N}\sum_{i=1}^{N}\frac{(accurate)_i}{|Z_i|}$$

where $(accurate)_i$ is a 0-1 variable, showing whether the classifier is accurate or not on the i-th instance; $|Z_i|$ is the number of classes returned on the i-th instance and N is the number of instances of the test set. However, discounting *linearly* the accuracy on the output size is arbitrary. For example, one could instead discount on $|Z_i|^2$.

The non-parametric *rank test* overcomes this problem. On each instance we rank two classifiers CL_1 and CL_2 as follows:

- if CL_1 is accurate and CL_2 inaccurate: CL_1 wins;
- if both classifiers are accurate but CL_1 returns less classes: CL_1 wins (the same for CL_2);
- if both classifiers are wrong: tie.
- if both classifiers are accurate with the same output size: tie;

We assign rank 1 to the classifier which wins, rank 2 to the classifier which looses and rank 1.5 to both classifiers in case of tie. Then, we check via the Friedman test (significance level 5%) whether the difference between the rank of the classifiers is significant. The rank test is more robust than d-acc, as it does not encode an arbitrary function for the discounting; yet, it uses less pieces of information and can therefore be less sensitive. Overall, a cross-check of the both indicators is recommended.

7.1. Overall performance of TANC

In this section, we evaluate the performance of TANC on *complete* data sets (missing data have been replaced by the mode for categorical variables, and by the average for numerical ones). On each training-test split of cross-validation, we learn the TAN structure using an algorithm (imported from the WEKA library [20]) which minimizes the MDL cost function.

In Figure 4, we present the boxplots (whose population is constituted by the results measured on 45 data sets) of three indicators of performance for TANC. TANC has a quite high determinacy (median around 90%); the determinacy generally increases with the number of instances (large data sets reduce the importance of the prior) and decreases with the number of classes and features. For example, we have taken the *kr-kp* data set (3196 instances) and observed monotonically increasing determinacy if we choose, from the 3196 instances, random subsets with 50, 100, 500 and 1000 instances to process (instead of all 3196 instances). The average determinacies (ten random runs for each subset size) are respectively 36%, 75%, 95% and 98%, which

[6]The metric is referred to as *precision* in [19].

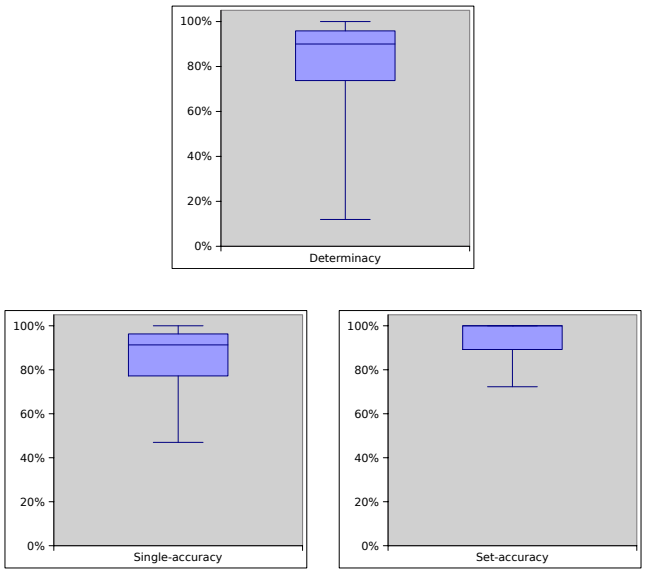

Figure 4: Boxplots of several performance indicators for TANC; the boxplots of determinacy and single-accuracy are computed on 45 data sets; the boxplot of set-accuracy is computed on 31 data set (the 14 binary data sets have been not considered).

support the expected theoretical behavior of as more determinacy as more data. Using a fixed joint distribution generating the data, probability intervals shrink with amount of data, and so determinacy increases. Yet, the speed of increase of determinacy with data depends on the data distribution: for instance, if a data set is generated from a very skewed/uneven distribution, determinacy will increase much slower.

We have observed very low determinacy on data sets which are small and contain many classes or features; for instance, the determinacy is under 20% on *audiology* (226 instances, 69 features, 24 classes), *primary-tumor* (339 instances, 17 features, 22 classes) and *squash-stored* (52 instances, 11 features, 3 classes). Such data sets require the estimation of a considerable number of parameters (because of the amount of joint states of features and classes) with a limited sample size; yet, the learned TAN structures do not seem to be aware of this problem, as they assign a second parent (besides the class) to most features (in principle, the TAN structure can assign *or not* a second parent to each feature). On audiology, a conditional density $p(F_1|f_2, c)$ (where the pair f_2, c denotes the joint values of the parents) contains generally 2 parameters, estimated on less than 5 instances. A similar situation is also found on the other mentioned data sets. Also the case of *optdigits* (5600 instances, 62 features, 10 classes, 10 states per feature) is interesting; despite the large size of the data set, TANC achieves a determinacy of only around 57%. In fact, a generic conditional density $p(F_1|f_2, c)$ contains on average 10 parameters, estimated on about 50 instances. Yet, since the features have uneven distributions, some densities are estimated on just 10-15 samples. The joint frequencies induced in the contingency tables are numerically small, causing the indeterminacy of TANC. In a modified version of the data set, where we have made

all features binary, the determinacy of TANC rises up to 98%. The reason for the indeterminacy of TANC on such data sets is therefore a *too complex* TAN structure with respect to the amount of data.

The boxplots about single-accuracy and set-accuracy show that TANC is reliable (medians are about 90% and 100%, respectively). The set-accuracy is especially high, showing that indeterminate classifications do preserve the reliability of TANC on hard-to-classify instances. On average, TANC returns about 70% of the classes of the problem, when it is indeterminate (excluding binary data sets from the computation).

7.2. TANC and TAN

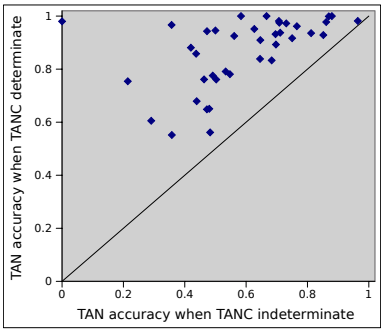

Figure 5: Scatter plot of the accuracies achieved by TAN on the instances determinately and indeterminately classified by TANC; each point refers to a data set.

We start the comparison between TANC and TAN by pointing out that the accuracy of TAN drops on average of 28 points on the instances which are indeterminately classified by TANC. In Figure 5 we present a scatter plot (each point refers to a data set) of the accuracy achieved by TAN on the instances classified determinately and indeterminately by TANC. In the following, by "decrease" we mean the decrease of TAN accuracy between instances which are determinately and indeterminately classified by TANC. A very small decrease is observed on *solar-flare-X* (98% to 96.5%); this is due to the fact that the majority class covers 98% of the instances, which can be seen as baseline for accuracy on this data set. Other data sets where the decrease is quite small include for instance *squash-unstored* (93% to 85%) and *grub-damage* (56% to 48%); such data sets have small number of instances with high number of classes or features; under such situations, as we have already seen, the structures are too complex and cause TANC to become excessively imprecise. Interestingly, on *optdigits* the decrease is from 99% to 86% on the original data set, but from 94% to 47% on the binary version. Otherwise, on data sets with two classes, the accuracy of TAN on the instances indeterminately classified is comparable to random guessing or even worse (diabetes: 9%; credit: 55%, kr-kp: 40%); however, as the number of classes increases, TAN performs better on the instances indeterminately classified; this might show that as the number of classes increases, as TANC is more unnecessarily indeterminate.

7.3. TANC and TANC*

Two main differences exist between TANC and TANC*: the model of prior ignorance (TANC adopts the EDM, while TANC* the local IDM) and the treatment of missing data (TANC* assumes MAR, while TANC assumes nonMAR). We focus on assessing the impact of the model of prior ignorance; to remove the effect of missing data, we work on *complete* data sets. We did not implement TANC* in our code; rather, we have compared our results with those published in [14]; although the comparison has to be taken with some cautiousness, the results underlie clear patterns which allow us to draw some conclusions.

We consider the six complete data sets analyzed in [14]. TANC is more determinate

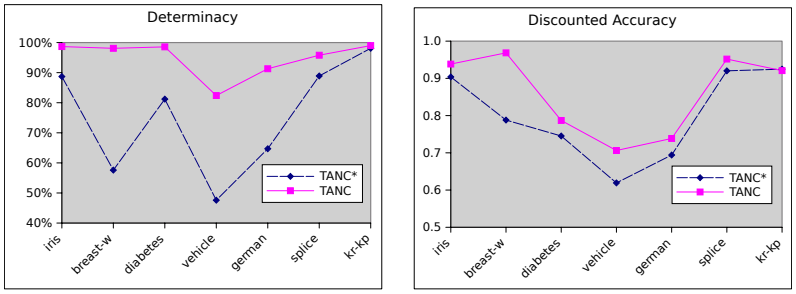

Figure 6: TANC vs TANC*.

than TANC*, as it can be seen from Figure 6; on average, the increase of determinacy is of 19 points percentile. This is the result of the smaller credal set built by the EDM compared to the local IDM. However, the determinacy of the two classifiers is equivalent on kr-kp (around 3200 instances, only binary features), where the role of the prior is not influential.

Moreover, from the indicators reported in [14], we build an approximate estimate of the discounted accuracy of TANC*, as follows:

$$\text{d-acc} \approx determ * singleAcc + \frac{setAcc * (1 - det)}{(indOutSz)} \tag{10}$$

where the first term is the contribution to d-acc coming from determinately classified instances, and the second term is the contribution from indeterminately classified ones. The approximation lies in the fact that, for the indeterminately classified instances, we divide the *average* accuracy by *indOutSz*, which is the *average* output size, instead of dividing accuracy and output size instance by instance and averaging only at the end over all the instances.

The d-acc (computed in the approximated way for both classifiers, to provide a fair comparison) are shown in Figure 6; TANC outperforms sensibly TANC* in all data sets, apart from kr-kp, where both classifiers perform the same.

7.4. TANC vs. NCC

Overall, TANC is slightly inferior to NCC, as shown by the scatter plot of the discounted accuracies of the two classifiers (Figure 7). The rank test returns 30 ties,

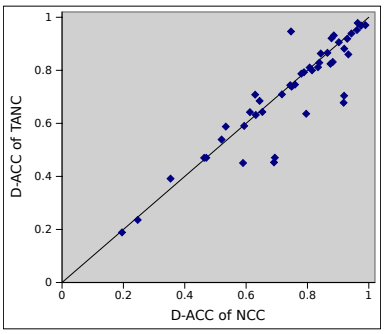

Figure 7: Scatter plot of the d-acc of TANC and NCC.

9 wins for NCC and 6 wins for TANC. The data sets where the rank test returns a victory of NCC include some data sets where the Bayesian TAN is outperformed by NBC (for instance, the already mentioned *labor, contact-lenses, pasture*) and which are in general quite small; however, they also include some further data sets where TAN is as good as, or even better than, NBC. A striking example is the already mentioned *optdigits*: here TAN is slightly better than NBC; their average accuracies are 94% and 92%. On this data set, NCC has determinacy 96%, and d-acc of 0.91; on the other hand, TANC has determinacy only of 57% (the reasons have been already analyzed), achieving a d-acc of 67%. The same pattern (lower d-acc of TANC due to much lower determinacy than NCC) is observed for instance also on *lymph*.

On the other hand, TANC outperforms NCC on data sets which include correlated variables; for instance, *kr-kp*, *vowel*, *monks-1*. Moreover, TANC outperforms NCC on the binary version of optdigits.

We recognize that the current implementation of TANC is generally less effective than NCC (although in a few examples it does outperform NCC), although the Bayesian TAN is generally more accurate than NBC. As already explained, the problem lies in the learned TAN structures, which should be simpler (i.e., contain less dependencies) to be better suited for a classifier based on imprecise probabilities. Currently, to our knowledge there are no structure learning methods that are specially designed for credal networks.

7.5. Preliminary results with missing data

In this section we compare the determinacy and the accuracy of TANC and NCC (in its updated version that is able to deal with nonMAR missing data [13]) in the presence of missing data. We recall that by nonMAR we indicate ignorance about the MP, which also implies that MAR cannot be assumed. We consider four data sets, whose characteristics are described in the first four column of Table 2. We considered the complete data sets and then artificially generate missing values by using a selective MP that targeted only certain values of the features, that is, for a given feature, we have randomly selected one of its categories and then removed (at random over the instances that contained that category) some of them. Such procedure leads to nonMAR missing

data. The data are divided into training and testing sets with a 2/3 split (testing set is complete as we only generate missing values in the training data).

	Number of				Determinacy (%)		D-ACC (%)	
Dataset	Feats	Inst.	Classes	Missing	TANC	NCC	TANC	NCC
breast-w	9	350	2	8	97.4	99.4	96.1	96.8
crx	15	345	2	54	84.9	90.1	85.2	84.9
soybean	35	290	19	128	00.0	00.0	13.9	15.7
vote	3	53	2	16	33.6	82.9	66.3	66.5

Table 2: Comparison of TANC and NCC in a few data sets with missing data.

As shown in Table 2, the determinacy of TANC is constantly inferior to that of NCC, and decreases drastically in the *vote* data set. In particular, this data set contains several instances where two features, which are interconnected in the TAN structure, are missing at the same time; this can explain the higher determinacy of NCC compared to TANC. On the other hand, the discounted accuracy of TANC in the very same data set remains equivalent to that of NCC, which shows that TANC was more accurate on deciding which instances are harder (or easier) to classify. In fact, using the *vote* data set, TANC has 98.63% of accuracy when it returns a single class, while NCC achieves only 70%. TANC is also more accurate when answering a single class on *breast-w* and *crx* (in the *soybean* data set, none of them ever answered a single class). This observation can also be concluded from the fact that TANC was slightly less determinate in all data sets, yet keeping the d-acc at the same level. TANC achieves slightly better results in the *crx* data set, slightly worse results in the *soybean* data set, and mostly the same accuracy in *breast-w* and *vote*. As already discussed in this section, data sets like *soybean* with many classes and features (when compared to the amount of data) are more susceptible to indeterminate classifications.

8. Conclusions

TANC is a new credal classifier based on a Tree-Augmented Naive structure; it treats missing data conservatively by considering all possible completions of the training set, but avoiding an exponential increase of the computational time. TANC adopts the EDM as a model of prior ignorance; we have shown that EDM is a reliable and computationally affordable model of prior near-ignorance for credal classifiers. We have shown that TANC is more reliable than precise TAN (learned with uniform prior) and that it obtains better performance compared to a previous TAN model based on imprecise probabilities, but learned with a local IDM approach; the adoption of EDM overcomes the problem of the unnecessary imprecision induced by the local IDM, while keeping the computation affordable.

TANC has shown good accuracy when compared to NCC, but overall is still behind NCC's performance. One main reason for such results lies on the algorithm to learn the TAN structure. Finding the best TAN structure is a challenging problem, and has strong impact even for precise classifier. In the case of credal classifiers such as TANC, the

structure must be learned accordingly, that is, the structure learning method must take into account the imprecise nature of the classifier to build the best structure for such model. Our experiments indicate that a more cautious structure with respect to that learned for a precise classifier might obtain better performance in the credal version. Unfortunately, learning the structure of a credal network is a hard problem currently without practical solutions, and we were forced to learn the structure using a standard method that does not take the credal nature of the model into account.

The TANC classifier has room for many improvements. The treatment of MAR and nonMAR missing data all together, appearing both in the training and the testing set are the main topics for future work. In order to make TANC less indeterminate on incomplete data sets, a solution could be to allow for mixed configurations, in which some features are treated as MAR and some others are not. This would allow both for a decrease of indeterminacy and for a finer-grained tuning of the way that missing data are dealt with. Besides that, the computational performance of TANC can also be further improved, for example, with the use of dynamic programming. Extensions beyond trees are also of interest, but they fall into the need of fast and accurate inference methods for general credal networks.

A further open problem, of interest in general for credal classification, is the development of metrics to compare credal classifier and classifiers based on traditional probability.

Acknowledgments

Work partially supported by the Swiss NSF grants n. 200021-118071/1 and n. 200020-116674/1 and by the project *Ticino in rete*.

References

[1] P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning 29 (2/3) (1997) 103–130.

[2] D. Hand, K. Yu, Idiot's Bayes-Not So Stupid After All?, International Statistical Review 69 (3) (2001) 385–398.

[3] J. Friedman, On bias, variance, 0/1 - loss, and the curse-of-dimensionality, Data Mining and Knowledge Discovery 1 (1997) 55–77.

[4] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian Network Classifiers, Machine Learning 29 (2) (1997) 131–163.

[5] M. Madden, On the classification performance of TAN and general Bayesian networks, Knowledge-Based Systems 22(7) (2009) 489–495.

[6] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, New York, 1991.

[7] J.-M. Bernard, An introduction to the imprecise Dirichlet model for multinomial data, International Journal of Approximate Reasoning 39 (2-3) (2005) 123–150.

[8] M. Zaffalon, Statistical inference of the naive credal classifier, in: G. de Cooman, T. L. Fine, T. Seidenfeld (Eds.), ISIPTA '01: Proc. of the Second International Symposium on Imprecise Probabilities and Their Applications, Shaker, The Netherlands, 2001, pp. 384–393.

[9] A. Cano, M. Gómez-Olmedo, S. Moral, Credal nets with probabilities estimated with an extreme imprecise Dirichlet model, in: ISIPTA '07: Proc. of the Fourth International Symposium on Imprecise Probabilities and Their Applications, SIPTA, Prague, 2007, pp. 57–66.

[10] R. J. A. Little, D. B. Rubin, Statistical Analysis with Missing Data, Wiley, New York, 1987.

[11] M. Zaffalon, Exact credal treatment of missing data, Journal of Statistical Planning and Inference 105 (1) (2002) 105–122.

[12] M. Zaffalon, Conservative rules for predictive inference with incomplete data, in: F. G. Cozman, R. Nau, T. Seidenfeld (Eds.), ISIPTA '05: Proc. of the Fourth International Symposium on Imprecise Probabilities and Their Applications, SIPTA, Manno, Switzerland, 2005, pp. 406–415.

[13] G. Corani, M. Zaffalon, Learning Reliable Classifiers from Small or Incomplete Data Sets: the Naive Credal Classifier 2, Journal of Machine Learning Research 9 (2008) 581–621.

[14] M. Zaffalon, E. Fagiuoli, Tree-Based Credal Networks for Classification, Reliable Computing 9 (6) (2003) 487–509.

[15] G. Corani, M. Zaffalon, Lazy naive credal classifier, in: Proc. of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data, ACM, 2009, pp. 30–37.

[16] G. Corani, M. Zaffalon, JNCC2: The Java Implementation Of Naive Credal Classifier 2, Journal of Machine Learning Research 9 (2008) 2695–2698.

[17] A. Asuncion, D. Newman, UCI machine learning repository, http://www.ics.uci.edu/∼mlearn/MLRepository.html (2007).

[18] U. M. Fayyad, K. B. Irani, Multi-interval Discretization of Continuous-valued Attributes for Classification Learning, in: Proc. of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1993, pp. 1022–1027.

[19] G. Tsoumakas, I. Vlahavas, Random k-Labelsets: An Ensemble Method for Multilabel Classification, in: Proc. of the 18th European conference on Machine Learning, Springer-Verlag Berlin, Heidelberg, 2007, pp. 406–417.

[20] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques (Second Edition), Morgan Kaufmann, 2005.

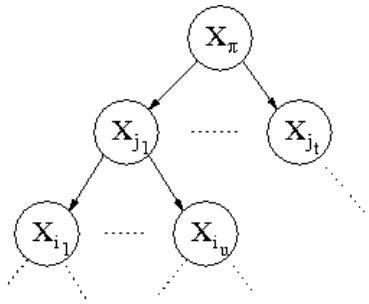

Figure A.8: Part of the computation tree of the TANC algorithm.

Appendix A. Correctness of the Algorithm for Dominance Test

This section describes the details and correctness of the algorithm to compute the value of the dominance test (Figure 3). The idea of the algorithm to evaluate Equation (9) is to combine the computations that are performed separately in the children of each variable and then to propagate the best possible solution to their sole parent. We ignore the arcs from C because we look for $\underline{p}(\mathbf{y}|c_1) = \min_{\mathbf{d}_{\mathcal{Y}}^{c_1}} p(\mathbf{y}|c_1)$ and $\overline{p}(\mathbf{y}|c_2) = \max_{\mathbf{d}_{\mathcal{Y}}^{c_2}} p(\mathbf{y}|c_2)$, that is, the actual root variable C is observed. The computation starts on the leaves and follows in a bottom-up idea. At each variable X_i, the goal is to obtain the joint probability $p(\mathbf{y}_{\sigma(\mathbf{i})}|y_i, c)$ of its descendants conditional on y_i[7] (c equals c_1 or c_2 depending whether it is the minimization or the maximization). This evaluation is done for all possible completions $\mathbf{d}_{X_i}^{c_1}$ and it is optimized over the completions of the children. The result is stored in a *cache* $\phi_i(\mathbf{d}_{X_i}^{c_1})$. Figure A.8 shows part of a network. At X_{j_1}, the joint probabilities $p(\mathbf{y}_{\sigma(\mathbf{i_k})}|y_{i_k}, c)$ of every child $X_{i_k} \in \Lambda_{j_1}$ (for every possible completion of that sub-tree) are already computed. So, they are combined to obtain $p(\mathbf{y}_{\sigma(j_1)}|y_{j_1}, c)$, for every possible completion of X_{j_1}. We perform such idea for each $j_1, \ldots, j_t$, obtaining the probabilities $p(\mathbf{y}_{\sigma(\mathbf{j_1})}|y_{j_1}, c), \ldots, p(\mathbf{y}_{\sigma(\mathbf{j_t})}|y_{j_t}, c)$ that are then made available to the parent X_π, where the computations are analogous but using the information obtained from X_{j_1} and its siblings. The process goes through the tree structure until reaching the root.

Suppose that the root variables (if C is not considered) are $X_1, \ldots, X_r$. So,

$$\begin{aligned} p(\mathbf{y}|c_1) &= \prod_{j=1}^{r} \left(p(y_j|c_1) \cdot p(\mathbf{y}_{\sigma(\mathbf{j})}|y_j, c_1)\right) \\ &= \prod_{j=1}^{r} \left(p(y_j|c_1) \cdot \prod_{X_i \in \Lambda_j} p(y_i|y_j, c_1) p(\mathbf{y}_{\sigma(\mathbf{i})}|y_i, c_1) \right), \end{aligned}$$

[7] $y_i \in \Omega_{X_i}$ is used as the notation for the state of X_i that appears in the test instance.

and, in general, $\min_{\mathbf{d}_{\mathcal{Y}}^{c_1}} p(\mathbf{y}_{\sigma(\mathbf{j})}|y_j, c_1) =$

$$= \min_{\mathbf{d}_{\mathcal{Y}}^{c_1}} \left(\prod_{X_i \in \Lambda_j} p(y_i|y_j, c_1) p(\mathbf{y}_{\sigma(\mathbf{i})}|y_j, c_1) \right). \tag{A.1}$$

Now, when we complete the variable X_j, the children Λ_j have separable computations. They are separable because the counts n that appear in the children of X_j are independent of each other as they concern disjoint subsets of variables (the structure is a tree, so $\mathbf{X}_{\sigma(\mathbf{i})} \cap \mathbf{X}_{\sigma(\mathbf{i}')} = \emptyset$ for $X_i, X_{i'} \in \Lambda_j$, with $i \neq i'$ and $X_j = \Pi_i = \Pi_{i'}$.). The only dependent value is $n_{c_1 y_j}$, as it appears in the denominators of distinct children of X_j (it appears in the denominator of the estimation of each $p(y_i|y_j, c_1)$ in Equation (A.1)). However, $n_{c_1 y_j}$ is fixed as the problem is solved for every possible completion of X_j. Besides that, note that the terms $\boldsymbol{\alpha}$ are not present in this formulation because we treat them using the hidden missing instance. Hence, the overall computation can be decomposed as

$$\min_{\mathbf{d}_{X_j}^{c_1}, \mathbf{x}_{\sigma(\mathbf{j})}} p(\mathbf{y}_{\sigma(\mathbf{j})}|y_j, c_1) = \min_{\mathbf{d}_{X_j, X_i}^{c_1}} \prod_{X_i \in \Lambda_j} \left(p(y_i|y_j, c_1) \cdot \min_{\mathbf{d}_{X_i}^{c_1}, \mathbf{x}_{\sigma(\mathbf{i})}} p(\mathbf{y}_{\sigma(\mathbf{i})}|y_i, c_1) \right),$$

which is solved for each completion of X_j in a recursive formulation: for all $\mathbf{d}_{X_j}^{c_1}$,

$$\begin{aligned} \phi_j(\mathbf{d}_{X_j}^{c_1}) &= \min_{\mathbf{d}_{X_i}^{c_1}} \prod_{X_i \in \Lambda_j} \left(p(y_i|y_j, c_1) \cdot \min_{\mathbf{d}_{X_i}^{c_1}, \mathbf{x}_{\sigma(\mathbf{i})}} p(\mathbf{y}_{\sigma(\mathbf{i})}|y_i, c_1) \right), \\ &= \prod_{X_i \in \Lambda_j} \min_{\mathbf{d}_{X_i}^{c_1}} \left(p(y_i|y_j, c_1) \cdot \phi_i(\mathbf{d}_{X_i}^{c_1}) \right), \\ &= \prod_{X_i \in \Lambda_j} \min_{\mathbf{d}_{X_i}^{c_1}} \left(\frac{n_{c_1 y_i y_j}}{n_{c_1 y_j}} \phi_i(\mathbf{d}_{X_i}^{c_1}) \right), \end{aligned} \tag{A.2}$$

where $\phi_j(\mathbf{d}_{X_j}^{c_1})$ is assumed to be equal to one if X_j is a leaf. The maximization is analogous. An important fact in Equation (A.2) is that it is enough to keep the best possible solution for every completion of a variable without having to record all the completions of its descendants. This is valid because $n_{c_1 y_i y_j}$ is known when the completion $\mathbf{d}_{X_i}^{c_1}$ is given ($n_{c_1 y_j}$ was already fixed by the completion of X_j), so completions of variables in $\mathbf{X}_{\sigma(\mathbf{i})}$ are irrelevant for the minimization in Equation (A.2), and it is enough to have the best possible solution of the children $\phi_i(\cdot)$ for each $\mathbf{d}_{X_i}^{c_1}$.

Because of that, the algorithm is implemented in a bottom-up manner so as the ϕ's of children are available when a given variable is treated, which reduces the complexity of the method to be exponential in the number of missing values of only two variables (a variable and its parent) instead of all missing values. It is worth mentioning that in the last step of the algorithm, all the values $\phi_i(\mathbf{d}_{X_i}^{c_1})$ are computed for each variable X_i, $i \leq r$, that has only the class as parent. Finally, we obtain

$$\underline{p}(\mathbf{y}|c_1) = \phi_C(\cdot) = \prod_{X_i \in \Lambda_C} \min_{\mathbf{d}_{X_i}^{c_1}} \left(\frac{n_{c_1 y_i}}{n_{c_1}} \phi_i(\mathbf{d}_{X_i}^{c_1}) \right), \tag{A.3}$$

and similarly for the maximization. This final step returns the desired values $\overline{p}(\mathbf{y}|c_2)$ and $\underline{p}(\mathbf{y}|c_1)$, which are later multiplied by $p(c_2)$ and $p(c_1)$, respectively, to obtain the value of the dominance test.

Appendix B. Detailed results data set by data set

Dataset	N_f	n	N_c	*TANCC performance*				*TAN*	
				Det.	**Sg-Acc**	**SetAcc**	**Ind.Sz.**	**Tan-P**	**Tan-I**
audiology	69	226	24	11.9%	98.1%	98.0%	14.1	98.1%	70.7%
breast-w	9	683	2	98.1%	97.8%	100.0%	2.0	97.8%	86.1%
cmc	9	1473	3	91.5%	55.2%	81.2%	2.1	55.2%	35.8%
contact-lenses	4	24	3	41.7%	100.0%	100.0%	2.7	100.0%	58.3%
credit	15	1000	2	91.3%	76.1%	100.0%	2.0	76.1%	46.3%
credit-a	15	690	2	96.1%	88.1%	100.0%	2.0	88.1%	42.0%
diabetes	6	768	2	98.6%	79.1%	100.0%	2.0	79.1%	53.3%
ecoli	6	336	8	91.1%	85.8%	91.7%	3.9	85.8%	43.7%
eucalyptus	17	736	5	77.9%	65.0%	80.2%	2.3	65.0%	48.0%
glass	7	214	7	73.9%	76.1%	87.6%	3.6	76.1%	50.3%
grub-damage	8	155	4	58.7%	56.2%	86.4%	2.4	56.2%	48.3%
haberman	3	306	2	95.8%	75.4%	100.0%	2.0	75.4%	21.4%
heart-c	11	303	5	25.5%	96.2%	80.5%	4.1	96.2%	76.5%
heart-h	9	294	5	73.3%	89.3%	83.4%	4.1	89.3%	69.7%
hepatitis	17	155	2	91.5%	85.9%	100.0%	2.0	85.9%	95.8%
iris	4	150	3	98.7%	94.6%	100.0%	2.5	94.6%	50.0%
kr-kp	36	3196	2	99.0%	92.5%	100.0%	2.0	92.5%	56.1%
labor	11	57	2	75.3%	97.5%	100.0%	2.0	97.5%	70.8%
liver-disorders	1	345	2	100.0%	63.2%	n.a.	n.a.	63.2%	n.a.
lymph	18	148	4	17.9%	93.6%	91.3%	2.6	93.6%	81.2%
monks1	6	556	2	100.0%	94.6%	n.a.	n.a.	94.6%	n.a.
monks2	6	601	2	96.0%	64.8%	100.0%	2.0	64.8%	47.2%
monks-3	6	554	2	99.6%	98.0%	100.0%	2.0	98.0%	0.0%

Table B.3: Detailed results data set by data set of TANC and TANC (first 22 data sets). TAN-P and TAN-I indicate the accuracy of the Bayesian TAN when TANC is respectively determinate and indeterminate. Moreover, N_f denotes the number of features, n the number of instances and N_c the number of classes.

Dataset	N_f	n	N_c	*TANCC performance*				*TAN*	
				Det.	**Sg-Acc**	**SetAcc**	**Ind.Sz.**	**Tan-P**	**Tan-I**
nursery	8	12960	5	94.1%	93.8%	80.6%	2.0	93.7%	71.1%
optdigits	62	5620	10	57.2%	99.9%	99.7%	5.6	99.9%	87.0%
optdgtBinary	63	5620	10	97.9%	94.3%	80.5%	2.2	94.3%	47.2%
pasture	10	36	3	76.0%	96.7%	100.0%	2.6	96.7%	35.7%
primary-tumor	17	339	22	14.9%	67.9%	67.3%	9.1	67.9%	43.9%
segment	7	810	7	93.9%	95.1%	98.0%	2.5	95.1%	62.7%
sol-flare_C	10	323	3	81.1%	90.0%	90.7%	2.3	90.0%	87.6%
sol-flare_M	10	323	4	69.1%	93.2%	75.9%	2.6	93.2%	69.6%
sol-flare_X	10	323	2	80.6%	97.5%	100.0%	2.0	98.2%	96.4%
sonar	21	208	2	89.0%	90.9%	100.0%	2.0	90.9%	64.6%
spect	22	267	2	87.7%	83.3%	100.0%	2.0	83.3%	68.3%
splice	34	3190	3	95.8%	97.2%	99.6%	2.1	97.2%	73.1%
squash-st	11	52	3	15.2%	91.7%	100.0%	2.7	91.7%	75.0%
squash-unst	14	52	3	14.9%	92.9%	100.0%	2.7	92.9%	85.2%
tae	2	151	3	100.0%	47.0%	n.a.	n.a.	47.0%	n.a.
vehicle	18	846	4	82.4%	77.6%	89.1%	2.3	77.6%	49.1%
vowel	13	990	11	76.4%	78.1%	89.3%	2.7	78.1%	54.7%
waveform	19	5000	3	92.0%	83.9%	99.6%	2.0	83.9%	64.5%
wine	13	178	3	94.4%	100.0%	100.0%	2.2	100.0%	66.7%
yeast	7	1484	10	95.5%	60.5%	72.3%	3.1	60.5%	29.1%
zoo	16	101	7	74.8%	100.0%	100.0%	3.6	100.0%	88.0%

Table B.4: Detailed results data set by data set of TANC and TAN (last 23 data sets). TAN-P and TAN-I indicate the accuracy of the Bayesian TAN when TANC is respectively determinate and indeterminate. Moreover, N_f denotes the number of features, n the number of instances and N_c the number of classes.

Dataset	N_f	n	N_c	*TANC*		*NCC*		*RankTest*
				Det.	**D-acc**	**Det.**	**D-acc**	
audiology	69	226	24	11.9%	0.24	9.8%	0.25	NCC
breast-w	9	683	2	98.1%	0.97	100.0%	0.98	TIE
cmc	9	1473	3	91.5%	0.54	96.5%	0.52	TIE
contact-lenses	4	24	3	41.7%	0.64	66.7%	0.80	NCC
credit	15	1000	2	91.3%	0.74	96.9%	0.75	TIE
credit-a	15	690	2	96.1%	0.87	98.3%	0.87	TIE
diabetes	6	768	2	98.6%	0.79	99.7%	0.78	TIE
ecoli	6	336	8	91.1%	0.81	92.2%	0.83	TIE
eucalyptus	17	736	5	77.9%	0.59	75.5%	0.53	TANC
glass	7	214	7	73.9%	0.64	72.4%	0.65	TIE
grub-damage	8	155	4	58.7%	0.47	58.7%	0.47	TIE
haberman	3	306	2	95.8%	0.74	95.8%	0.74	TIE
heart-c	11	303	5	25.5%	0.39	20.4%	0.35	TIE
heart-h	9	294	5	73.3%	0.71	77.5%	0.72	TIE
hepatitis	17	155	2	91.5%	0.83	94.8%	0.84	TIE
iris	4	150	3	98.7%	0.94	98.0%	0.94	TIE
kr-kp	36	3196	2	99.0%	0.92	98.8%	0.88	TANC
labor	11	57	2	75.3%	0.86	90.0%	0.93	NCC
liver-disorders	1	345	2	100.0%	0.63	100.0%	0.63	TIE
lymph	18	148	4	17.9%	0.47	58.2%	0.69	NCC
monks1	6	556	2	100.0%	0.95	100.0%	0.75	TANC
monks2	6	601	2	96.0%	0.64	96.7%	0.61	TIE
monks-3	6	554	2	99.6%	0.98	100.0%	0.96	TIE

Table B.5: Comparison of TANC and NCC data set by data set (first 22 data sets). Det denotes determinacy and d-acc denotes discounted accuracy. Moreover, N_f denotes the number of features, n the number of instances and N_c the number of classes.

Dataset	N_f	n	N_c	*TANC*		*NCC*		*RankTest*
				Det.	**D-acc**	**Det.**	**D-acc**	
nursery	8	12960	5	94.1%	0.91	99.7%	0.90	TIE
optdigits	62	5620	10	57.2%	0.68	96.1%	0.92	NCC
optdgtBinary	63	5620	10	97.9%	0.93	98.8%	0.89	TANC
pasture	10	36	3	76.0%	0.82	81.7%	0.88	TIE
primary-tumor	17	339	22	14.9%	0.19	10.7%	0.20	TIE
segment	7	810	7	93.9%	0.92	95.7%	0.93	TIE
sol-flare_C	10	323	3	81.1%	0.80	85.4%	0.81	TIE
sol-flare_M	10	323	4	69.1%	0.75	71.1%	0.76	TIE
sol-flare_X	10	323	2	80.6%	0.88	93.2%	0.92	TIE
sonar	21	208	2	89.0%	0.86	96.6%	0.84	TIE
spect	22	267	2	87.7%	0.79	95.2%	0.79	TIE
splice	34	3190	3	95.8%	0.95	99.1%	0.96	TIE
squash-st	11	52	3	15.2%	0.45	46.8%	0.59	NCC
squash-unst	14	52	3	14.9%	0.45	46.9%	0.69	NCC
tae	2	151	3	100.0%	0.47	92.7%	0.46	TIE
vehicle	18	846	4	82.4%	0.71	93.3%	0.63	TANC
vowel	13	990	11	76.4%	0.68	76.6%	0.64	TANC
waveform	19	5000	3	92.0%	0.81	99.3%	0.81	TIE
wine	13	178	3	94.4%	0.97	97.8%	0.99	TIE
yeast	7	1484	10	95.5%	0.59	97.0%	0.59	TIE
zoo	16	101	7	74.8%	0.83	80.6%	0.88	NCC

Table B.6: Comparison of TANC and NCC data set by data set (last 23 data sets). Det denotes determinacy and d-acc denotes discounted accuracy. Moreover, N_f denotes the number of features, n the number of instances and N_c the number of classes.