Lazy Naive Credal Classifier

Giorgio Corani
IDSIA
6928 Manno, Switzerland
giorgio@idsia.ch

Marco Zaffalon
IDSIA
6928 Manno, Switzerland
zaffalon@idsia.ch

ABSTRACT

We propose a local (or lazy) version of the *naive credal classifier*. The latter is an extension of naive Bayes to imprecise probability developed to issue reliable classifications despite small amounts of data, which may then be carrying highly uncertain information about a domain. Reliability is maintained because credal classifiers can issue set-valued classifications on instances that are particularly difficult to classify. We show by extensive experiments that the local classifier outperforms the original one, both in terms of accuracy of classification and because it leads to stronger conclusions (i.e., set-valued classifications made by fewer classes). By comparing the local credal classifier with a local version of naive Bayes, we also show that the former reliably deals with instances which are difficult to classify, unlike the local naive Bayes which leads to fragile classifications.

Keywords

Lazy Credal Classifier, Naive Credal Classifier, Imprecise Probabilities

1. INTRODUCTION

In a recent paper [3] we have argued that there is a 'dimension' of uncertainty in commonly available data for pattern classification that is often neglected by traditional classification methods. This kind of uncertainty arises because in data mining applications, data are most often the only source of knowledge about a domain: as a consequence, the mechanism generating the data is unknown a priori, that is, before looking at the data. The way a classifier deals with this prior ignorance can notably affect the conclusions it draws, in particular, it can lead it to draw fragile conclusions. We have shown that this often happens with Bayesian classifiers; the reason is that they model prior ignorance by *non-informative* priors,[1] which are questionable models of ignorance [12, Section 5.5.1].

At the same time we have proposed a generalization of Bayesian classifiers, to better deal with the introduced issue, that we have called *credal classifiers*. In particular, *naive credal classifier* (NCC) [3] extends *naive Bayes classifier* (NB) by modeling ignorance[2] by a set of priors (also called prior *credal set*).[3] This is turned into a set of posteriors by element-wise application of Bayes rule, and the classification is eventually issued by discarding from the set of classes those which are suboptimal for *all* the posteriors in the set (we call these classes *dominated*[4]). This leads NCC to return a set of classes when faced with instances whose classification would be prior-dependent for NB; it issues hence a weaker and yet arguably more robust classification than NB.

Moreover, by issuing a set of classes (we call this an *indeterminate* classification) on the instances whose classification is prior-dependent, NCC distinguishes the *hard-to-classify* instances (those for which the learning set is not informative enough, and that are thus prior-dependent) from the *easy-to-classify ones*, thus clearly bringing to light the dimension of data uncertainty mentioned initially. In other words, NCC automatically detects different degrees of uncertainty in the data and acts accordingly by issuing a more careful classification when an instance is recognized as difficult relative to the learning set. Experiments have shown that NCC achieves high accuracy and that it indeed preserves reliability thanks to indeterminate classifications on the hard-to-classify instances.

Two aspects of NCC which can be improved are (i) its constitutive naive assumption which can be too simplicistic for dealing with complex domains and (ii) that in some cases NCC might become too indeterminate, returning a set-valued classification also on instances which could be otherwise correctly classified using a smaller set of classes (up to a single class).

In this paper we address the two issues above proposing a local (or lazy) version of NCC that we call *local naive credal classifier* (LNCC). A local classifier is inferred, each time a new instance has to be classified, from a subset of the learning data made of instances that are 'similar' to the

[1]These often coincide with the uniform prior of some of its variants

[2]More precisely, we focus on what Walley calls *near*-ignorance.

[3]A set of distributions, or credal set, is the basic modeling unit in the behavioural theory of *imprecise probability* developed by Walley [12].

[4]In other words, class c_i is said to dominate c_j if *for all* the posteriors densities it holds that the probability of c_i is larger than that of c_j.

KDD-WS'09, June 28 2009, Paris, France

instance to classify.

This helps addressing point (i) because working locally reduces the chance of encountering strong dependencies [7]; from a different viewpoint, NB is known to be a low-variance, high-bias classifier [8] and working locally can be understood as a way to decrease the bias of the classifier. Moreover, NB is naturally suited for local learning because it does not need many data; NCC has the same characteristics with the further advantage of being based on imprecise probability. In addition, we expect that working locally improves also the determinacy of NCC, thus addressing (ii): in fact, selecting instances similar to the one we want to classify is a way to select the part of the learning set that is more informative about such an instance, dropping the parts that are less so; and the amount of information in the learning set is tightly related to the determinacy of NCC.

One major issue in the development of a local classifier is the so-called *bandwidth selection*, that is, the choice of the number of instances to include in the local learning set. Usually the bandwidth is chosen via some kind of cross-validation and is then kept constant for all the instances. Yet, tuning the bandwidth for each instance to classify can lead to better performance; see for instance the case of [1] in regression.

Our goal in this paper is to design a query-by-query bandwidth selector, which improves accuracy by reducing bias and helps removing unnecessary indeterminacy. We do so by again exploiting the peculiar features of NCC. In particular, we keep on including instances in the local learning set until NCC starts issuing a determinate classification on the instance to classify (note that this is clearly favors removing indeterminacy), or until we reach the end of the learning set. The rationale behind this criterion is the following: we expect that the local learning set selected in this way works well because NCC is determinate when the learning data is judged informative enough to draw a strong conclusion, such as a determinate classification. In other words, the (in)determinacy can be intended as a way to evaluate the quality of the learning set relative to the specific instance to classify: if we reach the point of high quality, we refrain from including further data. If we do not reach that point, we exhaust the entire learning set and still keep reliability by issuing a set-valued classification.

We investigate the effect of the above choices by extensive experiments to compare LNCC with NCC. To this extent we also propose a new (and, to our knowledge, the first) method to empirically compare credal classifiers. This task is not easy in general just because a credal classifier can output more than one class, and this makes it hard to base the comparison on a simple performance index such as the predictive accuracy typically used with traditional classifiers.

Results on 36 data sets show that LNCC substantially outperforms NCC; it significantly reduces indeterminacy without worsening (often improving) accuracy; out of 36 data sets, and after having cross-checked the outcome of two different statistical tests, we concluded that 15 times LNCC wins, 20 times there is a tie between the two classifier and only once LNCC loses. These results appear to be very strong, showing that NCC is basically almost always dominated by LNCC both on the front of accuracy and determinacy.

To strengthen our results, we have also applied our bandwidth selector to the inference of a local NB, in order to see more clearly whether the bandwidth selector provides advantages on its own. And indeed, the bandwidth selector is viable also for local naive Bayes, whose performance approaches that of the more sophisticated *tree-augmented naive Bayes classifier* [9].

Finally, the availability of the local NB allows us to compare it with LNCC, and the results clearly show that the former issues fragile classifications on the instances that are hard according to LNCC. This shows, once more, that the naive credal classifier can recognize the different levels of uncertainty in the data and taking advantage of it to provide reliable classifications.

In the rest of the paper we introduce credal classification, and in particular, the naive credal classifier, in Section 2. Lazy learning is introduced in Section 3. The criterion for bandwidth selection is proposed in Section 4, while the new method to compare credal classifier is in Section 5. Finally, the experimental analysis is discussed in Section 6.

2. CLASSIFICATION WITH IMPRECISE PROBABILITIES.

IDENTIFICATION OF NON-DOMINATED CLASSES

- set Non-DominatedClasses := $\mathcal{C}$;
- for class $c_1 \in \mathcal{C}$
  - for class $c_2 \in \mathcal{C}$, $c_1 \neq c_2$
    - if c_2 is dominated by c_1, drop c_2 from Non-DominatedClasses and break the internal loop;
- return Non-DominatedClasses.

Figure 1: Identification of non-dominated classes via pairwise comparisons; $\mathcal{C}$ denotes the set of classes of the problem.

In this paper we adopt the 0-1 loss function. Under this assumption, a traditional probabilistic classifier returns the class with the highest probability on the basis of a uniquely computed posterior density. A credal classifier should instead manage a set of posterior densities (*credal set of posteriors*); class c_1 is told to *dominate* c_2 if the probability of c_1 is larger than the probability of c_2 for *all* the distributions in the credal set of posteriors. According to the optimality criterion of [12, Section 3.9.2], a credal classifier should return the *non-dominated* classes; they can be detected via pairwise comparisons, as shown in Figure 1.

If several non-dominated classes are found, the classifier returns an *indeterminate* (or *set-valued*) classification. Non-dominated classes are incomparable, i.e., there is no information in the model for ranking them. Summing up, credal classifiers drop the dominated classes as sub-optimal; in some cases, they express indecision by returning a set of optimal classes instead of a single optimal class.

2.1 Naive credal classifier.

NCC models a situation of prior ignorance by specifying a set of prior distributions; the set is formally defined by using Walley's imprecise Dirichlet model [12]. NCC updates each prior with the observed likelihood, via element-wise

application of Bayes' rule; in this way, the set of priors is turned into a set of posteriors (posterior credal set).

Let us denote the classification variable by C, taking values in the finite set $\mathcal{C}$, where the possible classes are denoted by lower-case letters. We have k features $A_1, \ldots, A_k$ taking generic values $a_1, \ldots, a_k = \mathbf{a}$ from the sets $\mathcal{A}_1, \ldots, \mathcal{A}_k$; the features are assumed to be *discrete*.

We denote by $\theta_{c,\mathbf{a}}$ the joint chance that $(C, A_1, \ldots, A_k)$ equals $(c, \mathbf{a})$, by $\theta_{a_i|c}$ the chance that $A_i = a_i$ conditional on c; similarly, we denote by $\theta_{\mathbf{a}|c}$ the chance of $(a_1, \ldots, a_k)$ conditional on c.

The naive credal classifier inherits from naive Bayes the ('naive') assumption of probabilistic independence of the attributes given the class:

$$\theta_{\mathbf{a}|c} = \prod_{i=1}^{k} \theta_{a_i|c}. \tag{1}$$

The implications of such an assumption are thoroughly discussed for instance in [5].

In this paper we assume missing data to be MAR (missing at random); this assumption allows one to ignore missing data both in the training set and in the instance to be classified. However, NCC has been extended to deal with non-MAR missing data in [3].

Let N be the total number of samples; let $n(c)$ and $n(a_i\,|c)$ be the observed frequencies of class c and of $(a_i\,|c)$. The likelihood function can be expressed as a product of powers of the theta-parameters:

$$L(\theta|\mathbf{n}) \propto \prod_{c\in\mathcal{C}} \left[\theta_c^{n(c)} \prod_{i=1}^{k} \prod_{a_i\in\mathcal{A}_i} \theta_{a_i|c}^{n(a_i|c)} \right], \tag{2}$$

where $\mathbf{n}$ denotes the vector of all the above frequencies. Observe that for all c and i, the observations satisfy the *structural constraints* $0 \leq n(a_i\,|c) \leq n(c)$, $\sum_c n(c) = N$ and $\sum_{a_i\in\mathcal{A}_i} n(a_i\,|c) = n(c)$.

2.2 The Imprecise Dirichlet Model.

We start by the traditional inference based on a single Dirichlet prior to then introduce the imprecise Dirichlet model (IDM).

The prior density is expressed similarly to the likelihood function, except that frequencies $n(\cdot)$ are replaced everywhere by $st(\cdot) - 1$,[5] where

- s is a *positive* real number which can be regarded as the number of *hidden samples*, in the common interpretation of conjugate Bayesian priors as additional sample units (the number can be fractional, though);
- $t(\cdot)$ can be regarded as the proportion of units of the given type; for instance, $t_{c'}$ is the proportion of hidden units having class c' in the hidden samples. In fact, coefficients $t(\cdot)$ in the prior density have the same role of frequencies $n(\cdot)$ in the likelihood.

We have the following expression for the prior density:

$$f\,(\theta\,|\mathbf{t}, s\,) \propto \prod_{c\in\mathcal{C}} \left[\theta_c^{st(c)-1} \prod_{i=1}^{k} \prod_{a_i\in\mathcal{A}_i} \theta_{a_i|c}^{st(a_i|c)-1} \right].$$

[5]The products $st(\cdot)$ are more traditionally denoted by $\alpha(\cdot)$.

By multiplying the prior density and the likelihood function, we obtain a posterior density of the same form as the prior, with $st(\cdot)$ replaced by $st(\cdot) + n(\cdot)$. Thus the posterior density for the theta-parameters is a product of independent Dirichlet densities.

After having observed the training set (summarized by the frequencies $\mathbf{n}$) and after having specified all the coefficients $\mathbf{t}$ of the prior density, we have:

$$P(c, \mathbf{a}|\mathbf{n}, s, \mathbf{t}) = P(c|\mathbf{n}, s, \mathbf{t}) \prod_{i=1}^{k} P(a_i|c, \mathbf{n}, s, \mathbf{t}) = \tag{3}$$

$$\frac{n(c) + st(c)}{N + s} \prod_{i=1}^{k} \frac{n(a_i\,|c) + st(a_i\,|c)}{n(c) + st(c)}. \tag{4}$$

The IDM allows the parameters $t(\cdot)$ to vary within certain intervals instead of being fixed to precise values. The structural constraints on the parameters $t(\cdot)$ are designed in analogy with those of the frequencies: $\sum_c t(c) = 1$, $\sum_{a_i} t(a_i\,|c) = t(c)$, $t(a_i\,|c) > 0\ \forall(i, c)$. As the IDM takes into consideration all the priors densities which satisfy the structural constraints, we can say that it models ignorance by the body of all possible states of knowledge.

For a given class c', the estimation of $P(c', \mathbf{a}|\mathbf{n}, \mathbf{t})$ via Eq. (4) under the IDM returns an interval, obtained varying the $t(\cdot)$ within their bounds; testing whether c' credal-dominates c'' means to verify whether:

$$\inf_{\mathbf{t}} \frac{p(c', \mathbf{a}|\mathbf{n}, \mathbf{t}, s)}{p(c'', \mathbf{a}|\mathbf{n}, \mathbf{t}, s)} > 1, \tag{5}$$

where the $t(\cdot)$ are subject to the structural constraints; the method to solve this optimization problem is given in [14].

3. LAZY CLASSIFIERS

Lazy classifiers defer the effort of learning until they are provided with an instance to classify (*query*); when this happens, (a) the instances of the training set are ranked according to the distance from the query; (b) a local classifier is induced on the k closest instances and is used to issue the classification. Eventually, (c) the local classifier is discarded, while the training set is kept in memory in order to answer future queries. Lazy classifiers are *local,* being fitted to a subset of instances in the neighborhood of the query-point.

Why is it interesting to locally learn naive Bayes (and, as a consequence, also NCC)? On the one hand, NB is a high-bias, low variance classifier (see [8] for a theoretical analysis and [11] for an experimental one) and local learning reduces the bias; on the other hand, local learning reduces the chance of encountering strong dependencies between features [7]; there is in fact evidence that local naive Bayes can outperform global naive Bayes [15, 7].

Probably, the most important parameters to be chosen for lazy learners is the number of neighbors k (*bandwidth*); the simplest approach is to choose the value of k which maximizes the accuracy measured by cross-validation on the instances of the training set; then, all queries are answered by using the same (optimized) k.

However, adapting the bandwidth query-by-query can significantly improve the performance of lazy learning, as shown, in the case of regression, in [1]; the framework of [1] is as follows: each time a query is received, several local linear regressors (each with different k) are identified; the leave-

one-out (loo) error[6] of each local regressor is measured and finally the local regressor with the lowest loo error returns the prediction. The algorithms of [1] are very efficient because they accomplish recursively most computations.

The selective neighborhood naive Bayes [15] adopts a similar approach for classification: when the query is provided, several local naive Bayes are induced (each using a different bandwidth), and the local classifier minimizing the loo error returns the classification. However, in [15] no particular techniques are adopted to speed up the computation, which is therefore very time-consuming.

Finally, in [7] the local naive Bayes is trained by assigning to each instance a weight that is inversely proportional to the distance from the query-point; thanks to this weighted learning, the local NB is quite robust to the choice of k, so that a general choice such as $k = 50$ or $k = 100$ leads to satisfactory results on most data sets. The locally weighed NB of [7] is shown to be better than NB, and be competitive against more sophisticated Bayesian classifier such as tree-augmented networks or lazy Bayesian rules.

Local classifiers can be adversely affected by irrelevant features, which are accounted for in the distance computation and can bias the rank of the instances. Assigning weights to the features when computing the distance can be improve the performance of lazy classifiers (see [13] for a review); however, this out of the scope of this paper. In our experiments we use the simple overlap distance, i.e., the distance corresponds to the number of features that are different between the two compared instances.

4. CRITERION FOR BANDWIDTH SELECTION

Our bandwidth selector tunes the bandwidth query-by-query; it has an easy design and does not require much computational effort, unlike the loo criterion adopted in [15]. We work in an unweighted setting (or, in other words, all the instances have weight equal to one).

After having ranked the instances according to their distance from the query, a local NCC is induced on the k_{min} closest instances (we set $k_{min} = 25$) and classifies the instance. The classification is accepted if determinate; otherwise, LNCC is updated by adding a set of further k_{upd} instances (we set $k_{upd} = 20$) to its training set. The procedure continues until the classification is determinate or all instances of the training set has been used to learn. Therefore, the bandwidth is increased until the collected evidence is enough to smooth the effect of the choice of the prior; in fact, if the classification returned by LNCC is determinate, any local NB induced on the same data, whose prior is included in the IDM would return that very same classification.

The naive architecture makes it especially easy updating LNCC with the k_{upd} instances; it only requires to update the counts $n(\cdot)$ that are internally stored by LNCC.

Note that using the overlap distance can cause several instances to have the same distance from the query; in particular, if the instances k-th, $(k+1)$-th,.. ,$(k+m)$-th have the same distance from the query, the local classifier is induced on $k+m$ instances; for the same reason, it might happen that more than k_{upd} instances are selected to update LNCC.

[6]Leave-one-out is performed *locally,* i.e., the error is measured using only the instances used to train the local regressor and not the whole training set.

5. COMPARING CREDAL CLASSIFIERS

We refer to a classifier as *accurate* on a certain instance if its output includes the correct class, regardless how many classes it has returned; we refer to a classifier as *determinate* if its output contains only a single class.

In order to have a complete picture of the performance of a credal classifier, 4 indicators have been used in [3]: *determinacy*: i.e, the percentage of determinate classifications; *single accuracy*: the accuracy of the classifier when determinate; *set-accuracy*: the accuracy of the classifier when indeterminate; *indeterminate output size*: the average number of classes returned by the classifier when indeterminate. Each indicator measures a different aspect of the overall performance of a credal classifier; yet, for the sake of comparison, we need a more synthetic approach.

Since credal classifiers have been only recently introduced, there is not yet an established method for comparing two credal classifiers; in this paper, we make an effort in this direction. In particular, we consider two measures: (a) an indicator borrowed from multi-label classification and (b) a non-parametric test based on ranking the credal classifiers on each instance. From multi-label classification we import the *discounted-accuracy*:[7]

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

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. To understand how d-acc works in our setting, let us consider the following example: we have a test set containing $2m$ instances and classifiers CL_1 and CL_2; CL_1 classifies determinately and accurately m instances, but is inaccurate on the remaining m; CL_2 instead classifies accurately all the $2m$ instances, returning 2 classes on each instance. In this case, CL_1 is seen as equivalent to CL_2 by d-acc; yet, we would have drawn a different conclusion, if we had used $|Z_i|^2$ or $\sqrt{|Z_i|}$ at the denominator of d-acc. The point is that deciding *how* to discount the accuracy on the output size contains some unavoidable arbitrariness.

In order to address this problem, we introduce an alternative approach based on a rank test; in particular, on each instance we rank two classifiers CL_1 and CL_2 as follows:

- if CL_1 is accurate and CL_2 inaccurate: rank(CL_1) =1, rank(CL_1) =2 (and vice versa);
- if both classifiers are accurate but CL_1 returns less classes: rank(CL_1) =1, rank(CL_1) =2 (and vice versa);
- if both classifiers are wrong, rank(CL_1)=rank(CL_2)=1.5 (tie);
- if both classifiers are accurate and their output has the same size, rank(CL_1)=rank(CL_2)=1.5 (tie).

Eventually, we use the Friedman test (in particular, the open source implementation from SOCR[8] [4]) to check whether

[7]This indicator is referred to as precision in [10]; however, this term might be misleading in the setting of this paper.

[8]`http://www.socr.ucla.edu/`

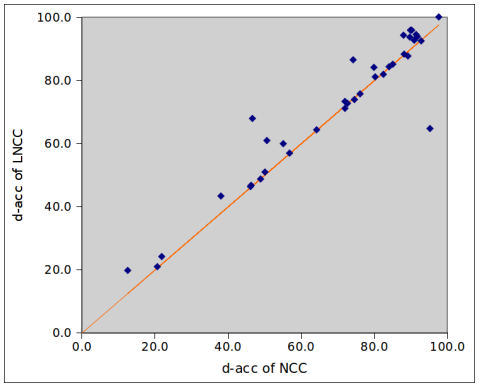

Figure 2: D-acc of NCC and LNCC.

the difference between the average rankings is significant. The rank test is more robust than d-acc, as it does not encode any (arbitrary) functional form for discounting accuracy on the basis of the output size; yet, it uses less pieces of information than d-acc and can be therefore be less sensitive. Overall, a cross-check of the results arising from both tests should allow drawing reliable conclusions.

6. EXPERIMENTS SETUP

We have compared LNCC and NCC on 36 data sets taken from the UCI repository; on each data set, we have performed 10 runs of 10-folds cross-validation. As in [3] we had analyzed the performance of NCC on 18 data sets only, part of these results are novel also for NCC.

For both LNCC and NCC, we have set s=1. Numerical features have been discretized via the entropy-based algorithm of [6] (the discretization intervals have been estimated on the training set, and then applied unchanged on the test set).

We have performed three types of experiments: (a) we have compared LNCC vs. NCC; (b) we have compared the *local naive Bayes* (LNB), i.e., a naive Bayes learned locally using the bandwidth decided by our criterion (see Section 7.1 for details), against naive Bayes and tree-augmented networks [9]; finally, (c) we have compared LNCC and LNB.

In this paper we do not address the comparison between NB and NCC; it has been shown in [3] with a large set of experiments that NCC is good at isolating hard-to-classify instances, as the accuracy of NB sharply drops the instances that are indeterminately classified by NCC. We will instead make a similar comparison between LNB and LNCC.

The indicators of performance generated by cross-validation on each data set have been pairwise compared via the t-test (α = 5%); unfortunately, the software we used does not implement a resampling-corrected t-test; therefore, the outcome of t-test should be taken with some caution, as it might give rise to some false positives.

We have implemented LNCC in Java; we plan to release it as open source software, as we already did with JNCC [2], i.e., the implementation of NCC.

7. RESULTS

In the following, by 'win' we mean that LNCC wins and by 'loss' we mean that LNCC loses.

The results data set by data set are shown in Table 1. Overall, the rank test reports respectively 15 wins, 19 ties and 2 losses; the analysis of d-acc (accomplished via t-test) reports instead 19 wins, 11 ties and 6 losses. A scatter plot showing d-acc of LNCC and NCC is given in Fig. 2. In the following, we review the results produced by the two tests.

LNCC wins (according to *both* tests) on 15 data sets; in 8 cases the improvement on d-acc is larger than 5 points; in 3 cases (out of the 8), the improvement is larger than 10 points.

On further 10 data sets, *both* tests return tie; these data sets have mostly a limited number of instances (e.g., labor, liver-disorder, lung-cancer, pasture, zoo, audiology) and in these cases LNCC uses in fact the entire training set to learn.

On 5 further data sets (balance, german-credit, hearth-statlog, ionosphere, iris) the t-test over d-acc reports a loss of LNCC; yet, on all such data sets the rank test returns instead a tie; moreover, the differences in d-acc are very thin (the largest, on ionosphere, is 1.6 points; in all the other cases, the difference is smaller than 1 point); we conclude therefore that on these data sets there are 5 ties (we will later achieve similar conclusion also for some data sets where the t-test assigns a win to LNCC). On audiology, the rank test returns a victory of NCC; yet LNCC achieves a slightly higher value of d-acc; the point is that the two classifiers are very close to each other (there is a difference of less than one point on determinacy, accuracy and set-accuracy) and we conclude this is yet another tie. We evaluate as ties the outcomes on 4 further data sets (tae, primary-tumor, haberman and e.coli) where the t-test assigns a victory to LNCC but the rank test returns instead a tie and the difference in the value of d-acc is very small.

The only data set where LNCC is defeated is splice: the d-acc is 95% for NCC and 65% for LNCC; a more detailed inspection (see last row of Table 2) shows that the two classifiers have the same determinacy (around 99%) but NCC, when determinate, is 95% accurate; LNCC, when determinate, is only 65% accurate. The reason of the bad performance of LNCC is not clear. Feature selection shows that 28 features out of 60 are either redundant or irrelevant; removing these features increases of five points the d-acc of LNCC, which remains however far from NCC. Experiments with fixed bandwidth show that good performance can be achieved with bandwidth either smaller than 100 or larger than 2500; for unclear reasons, our criterion often picks a bandwidth from this (wide) sub-optimal region.

Eventually, we conclude that there are 15 wins for LNCC, 20 ties and 1 loss.

In Table 2 we report determinacy and single-accuracy of both classifiers on the 9 data sets where the absolute difference in d-acc exceeds 5 points; they include 8 data sets favorable to LNCC and the already analyzed splice; a more complete analysis would have required to consider also set-accuracy; yet, set-accuracy is generally very high for both classifiers and it is unlikely to be a reason of major differences in d-acc. Note that the column ΔDeterm of Table 2 contains only positive values, as LNCC cannot be less determinate than NCC.

On large data sets such as letter (20000 instances), nursery (13000 inst), pendigits (10000 inst), both NCC and LNCC have high determinacy; the improvement of LNCC is therefore due to a better single-accuracy; in these data cases, LNCC selects generally a bandwidth of only a few hundreds and the local estimator is much more accurate than

Data set	*Inst.*	*Classes*	*Feats.*	LNCC *d-acc(%)*	*Comparison with NCC* Δ*(d-acc)*	*rank Test*
anneal	898	6	38	67.9	**21.3**	W
audiology	226	24	69	21.0	0.2	L
balance-scale	625	3	4	71.2	**-0.8**	T
ecoli	336	8	7	81.0	**0.6**	T
eucalyptus	736	5	19	59.9	**4.6**	W
german_credit	1000	2	20	73.8	**-0.7**	T
grub-damage	155	4	8	46.2	0.1	T
haberman	306	2	3	73.4	**1.4**	T
heart-statlog	270	2	13	81.9	**-0.7**	T
hepatitis	155	2	19	84.3	0.2	T
ionosphere	351	2	34	87.7	**-1.6**	T
iris	150	3	4	92.5	**-0.3**	T
labor	57	2	16	88.2	0.1	T
letter	20000	26	16	86.5	**12.1**	W
liver-disorders	345	2	6	56.9	0.0	T
lung-cancer	32	2	56	64.2	0.0	T
mushroom	8124	2	22	100.0	**2.3**	W
nursery	12960	5	8	95.8	**5.6**	W
optdigits	5620	10	64	93.9	**1.9**	W
pasture-production	36	3	22	72.7	0.0	T
pendigits	10992	10	16	94.3	**6.3**	W
postoperative-patient-data	90	3	8	61.0	**10.3**	W
primary-tumor	339	22	17	24.1	**2.3**	T
segment	2310	7	19	92.7	**1.8**	W
solar-flare	323	2	12	94.5	**3.0**	W
sonar	208	2	60	75.7	-0.6	T
spambase	4601	2	57	93.6	**3.8**	W
splice	3190	3	60	64.7	**-30.6**	L
squash-stored	52	3	24	48.7	-0.3	T
squash-unstored	52	3	23	51.0	0.8	T
tae	151	3	5	46.7	**0.4**	T
vote	435	2	16	95.9	**5.8**	W
vowel	990	11	13	19.8	**7.3**	W
waveform	5000	3	40	84.0	**4.1**	W
white-clover	63	4	31	43.3	**5.3**	W
zoo	101	7	16	85.1	0.0	T

Table 1: Results data set by the data set; Δ(d-acc) is defined as (d-acc_{LNCC}) - (d-acc_{NCC}) and is boldfaced when the difference is statistically significant; for the rank test W means win of LNCC, T means ties and L means loss of LNCC.

	Baseline: NCC			LNCC variations		
Data set	determ(%)	singleAcc(%)	d-acc(%)	Δ(Determ)	Δ(SingleAcc)	Δ(d-acc)
anneal	0.7	100.0	46.7	40.8	0.0	21.3
letter	95.1	76.8	74.3	4.5	9.9	12.1
postop.-patient	47.7	63.9	50.7	18.9	12.0	10.3
vowel	0.0	n.a.	12.5	9.4	91.1	7.3
pendigits	97.9	89.1	88.0	1.8	5.4	6.3
vote	99.6	90.2	90.0	0.4	5.6	5.8
nursery	99.7	90.4	90.3	0.3	5.4	5.6
white-clover	9.2	94.3	38.0	10.5	-2.0	5.3
splice	98.6	96.0	95.3	1.1	-31.2	-32.2

Table 2: Analysis of the data sets where the absolute difference in d-acc between NCC and LNCC is larger than 5 points; NCC wins on splice, while LNCC wins on the remaining 8 cases.

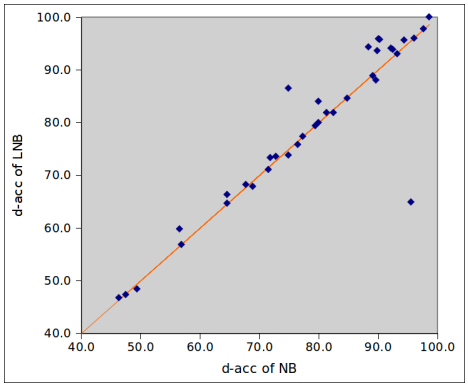

Figure 3: Scatter plot of accuracy for NB and LNB.

the global one; similar findings on large data sets can be found also in [7].

On data sets such as postoperative, white-clover and anneal there is instead a large improvement of determinacy, thanks to the peculiar design of the bandwidth selector. Remarkably, the increase in determinacy is accompanied also by an increase of single-accuracy, which means that the classifier is *both* more determinate and reliable. Only in the case of white clover there is a small decrease of single-accuracy; however, this is welcome, as LNCC increases the instances determinately classified from 10% to 20%, with very-high single accuracy of 92% (NCC achieves 94%); NCC is therefore more cautious than necessary, as LNCC discovers a new set of instances which can be classified very accurately. Let us moreover point out that LNB (which is better than NB) only achieves 62% accuracy on the instances that are classified indeterminately by LNCC; this a good evidence that LNCC sensibly isolates hard-to-classify instances.

7.1 Local Naive Bayes.

In this section we introduce the local naive Bayes, i.e., a naive Bayes which is locally learned using the bandwidth selected by our criterion. In fact, although our criterion is tailored to the imprecise probabilities setting, it should allow LNB to issue reasonably good classifications. First, we compare LNB against NB; a scatter plot of the accuracies of the two classifiers is given in Fig. 3 and suggests a better overall performance of LNB (with the only exception of splice, which can be easily spotted). The outcome of the t-test is 16 wins of LNB , 13 ties and 7 wins of NB. On 4 data sets LNB wins with more than 5 points of margin over NB; on the other hand, the maximum advantage of NB over LNB is 1.5 points (apart from splice). We can therefore conclude that overall LNB outperforms NB.

In fact, LNB approaches the performance of tree-augmented networks (TAN), which have been shown [9] to clearly outperform NB: the comparison shows 10 wins for LNB, 15 ties and 11 wins for TAN. On 5 data sets, the difference in accuracy is larger than 5 points: 2 times in favor of LNB and 3 times (including splice) in favor of TAN.

We finally assess the ability of LNCC of isolating hard-to-classify from easy-to-classify instances. To this purpose, we measure the accuracy of LNB (which is the Bayesian counterpart of LNCC) on the instances classified determinately and indeterminately by LNCC; if LNCC is good at isolating hard-to-classify instances, the accuracy of LNB should drop on the instances indeterminately classified by LNCC. We do not carry a deep analysis of these results; however, it should suffice to say that on the average the accuracy of LNB drops of about 22 points between the instances classified determinately and indeterminately by LNCC.

8. CONCLUSIONS

In this paper we have proposed LNCC, which is a local version of the naive credal classifier. The latter is an extension of naive Bayes to imprecise probabilities, which is suited to isolate different degrees of uncertainty in the data and to exploit this finding in order to issue robust classifications. Robustness is obtained on difficult cases by classifying an instance using a set of classes, rather than a single class. We call this an indeterminate classification. Extensive experiments show that LNCC achieves better overall performance than NCC; moreover, we have shown that it is indeed capable of issuing reliable classification by comparing it with a related local version of naive Bayes: the latter is show to be unreliable when LNCC is not able to classify an instance by a single class. These key features of LNCC have been achieved by relying on an original way to select the bandwidth of the local classifier that relies specifically on the key feature of credal classifiers to distinguish hard-from easy-to-classify instances.

Regarding future research, one important avenue would be to extend the present treatment to missing (or incomplete) data. In fact, missing data introduce another difficult degree of data uncertainty that NCC has been shown to cope with reliably, too, using the paradigm of imprecise probability.

However, the problem of bandwith selection is a difficult one and selecting a unique bandwidth might be in some cases a fragile approach; a model averaging framework for local estimators could instead overcome this problem, as shown for the case of regression in [1]. Model averaging for credal classifiers has been not yet investigated and we regard it as an interesting subject of research.

Acknowledgments

Work partially supported by the Swiss NSF grants n. 200021-118071/1 and 200020-116674/1.

9. REFERENCES

[1] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least squares algorithm. *Advances in Neural Information Processing Systems*, pages 375–381, 1999.

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

[3] G. Corani and M. Zaffalon. Learning reliable classifiers from small or incomplete data sets: The naive credal classifier 2. *Journal of Machine Learning Research*, 9:581–621, 2008.

[4] I. D. Dinov. Socr: Statistics online computational resource. *Journal of Statistical Software*, 16(11):1–16, 10 2006.

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

[6] U. M. Fayyad and K. B. Irani. Multi-interval Discretization of Continuous-valued Attributes for

Classification Learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, San Francisco, CA, 1993. Morgan Kaufmann.

[7] E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive Bayes. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

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

[9] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian networks classifiers. *Machine Learning*, 29(2/3):131–163, 1997.

[10] G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Proceedings of the 18th European conference on Machine Learning*, pages 406–417. Springer-Verlag Berlin, Heidelberg, 2007.

[11] P. Van Der Putten and M. Van Someren. A bias-variance analysis of a real world: the CoIL challenge 2000. *Machine Learning*, 57:177–195, 2004.

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

[13] D. Wettschereck, D. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1):273–314, 1997.

[14] M. Zaffalon. Statistical inference of the naive credal classifier. In G. de Cooman, T. L. Fine, and T. Seidenfeld, editors, *ISIPTA '01: Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 384–393, The Netherlands, 2001. Shaker.

[15] X. Zhipeng, W. Hsu, L. Zongtian, and M. Lee. SNNB: A Selective Neighborhood Based Naïve Bayes for Lazy Learning. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 104–114. Springer-Verlag London, UK, 2002.