Information Sciences 181 (2011) 2656–2670

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Calibrated lazy associative classification

Adriano Veloso [a,*], Wagner Meira Jr. [a], Marcos Gonçalves [a], Humberto M. Almeida [a], Mohammed Zaki [b]

[a] Computer Science Dept., Federal University of Minas Gerais, Brazil
[b] Computer Science Dept., Renseelaer Polytechnic Institute, USA

ARTICLE INFO

Article history:
Received 10 February 2009
Received in revised form 7 March 2010
Accepted 10 March 2010
Available online 20 March 2010

Keywords:
Classification
MDL
Calibration

ABSTRACT

Classification is a popular machine learning task. Given an example x and a class c, a classifier usually works by estimating the probability of x being member of c (i.e., membership probability). Well calibrated classifiers are those able to provide accurate estimates of class membership probabilities, that is, the estimated probability $\hat{p}(c|x)$ is close to $p(c|\hat{p}(c|x))$, which is the true, (unknown) empirical probability of x being member of c given that the probability estimated by the classifier is $\hat{p}(c|x)$. Calibration is not a necessary property for producing accurate classifiers, and, thus, most of the research has focused on direct accuracy maximization strategies rather than on calibration. However, non-calibrated classifiers are problematic in applications where the reliability associated with a prediction must be taken into account. In these applications, a sensible use of the classifier must be based on the reliability of its predictions, and, thus, the classifier must be well calibrated.

In this paper we show that lazy associative classifiers (LAC) are well calibrated using an MDL-based entropy minimization method. We investigate important applications where such characteristics (i.e., accuracy and calibration) are relevant, and we demonstrate empirically that LAC outperforms other classifiers, such as SVMs, Naive Bayes, and Decision Trees (even after these classifiers are calibrated). Additional highlights of LAC include the ability to incorporate reliable predictions for improving training, and the ability to refrain from doubtful predictions.

1. Introduction

The classification problem is defined as follows. We have an input data set called the *training data* (denoted as $\mathcal{D}$) which consists of examples (or instances) composed of a set of m attribute-values $(a_1, a_2, \ldots, a_m)$ along with a special variable called the *class*. This class variable draws its value from a discrete set of classes $(c_1, c_2, \ldots, c_n)$. The training data is used to construct a classifier that relates the features (or attribute-values) in the training data to the class variable. The *test set* (denoted as $\mathcal{T}$) for the classification problem consists of a set of instances for which only the features are known while the class value is unknown. The classifier, which is a function that receives as input a set of features and outputs a class, is used to predict the class value for test instances.

There are countless paradigms and strategies for devising a classifier. One of these strategies is to exploit relationships, dependencies and associations between features and classes. These associations are usually hidden in the examples, and when uncovered, they may reveal important aspects concerning the underlying phenomenon that produced the examples.

* Corresponding author.
E-mail addresses: adrianov@dcc.ufmg.br (A. Veloso), meira@dcc.ufmg.br (W. Meira Jr.), mgoncalv@dcc.ufmg.br (M. Gonçalves), hmossri@dcc.ufmg.br (H.M. Almeida), zaki@cs.rpi.edu (M. Zaki).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.03.007

These aspects can be exploited for the sake of prediction. This is the strategy adopted by associative classifiers, where the classification function is built from rules [16] of the form $\mathcal{X} \rightarrow c_i$ (where $\mathcal{X}$ is a set of features and c_i is a class). Associative classifiers have been shown to be accurate in many applications, including document categorization [26], ranking for Information Retrieval [23] and bioinformatics [4]. There are other applications, however, where high accuracy is not the only requirement, not even the most important one, as illustrated by the following examples:

- Cautious classification – Consider a digital library in which documents must be sorted into pre-specified categories. The administrator may consider to use a classifier, as long as it matches a minimum acceptable accuracy (which is specified by the administrator). In order to fulfill this accuracy constraint, the classifier must provide estimates of class membership probabilities for each document x (i.e., $\hat{p}(c_1|x), \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$). These estimates are used to select the documents that are safely classified, and the documents that must have their corresponding predictions abstained. The estimates are also used to calculate the accuracy after each prediction. The classifier must stop when the estimated accuracy reaches its minimum acceptable value, and then the administrator performs manual inspection of the remaining documents. In this case, in addition to achieving high accuracy, the classifier must also inform which documents are likely to be correctly classified and those that are not (i.e., they will be manually classified).
- Cost-sensitive classification – Consider an organization which requests donations. Each request has a fixed cost z, and the expected amount that an individual x will donate is $y(x)$ dollars. The organization wants to improve its marketing efforts by developing a classifier that will be used to maximize its net revenue. According to [29], the optimal net maximization strategy is to solicit x if and only if $\hat{p}(\text{donate}|x) > \frac{z}{y(x)}$ (i.e., a marketer will include x in the mailing list only if the expected return from this order exceeds the cost invested in generating the order), where $\hat{p}(\text{donate}|x)$ is the estimated probability that x will donate. In this case, in addition to achieving high accuracy, the classifier must also take into account the risk of performing an incorrect prediction of donation.
- Learning to rank – Consider a state-of-the-art search engine which returns documents with different levels of relevance with regard to a query (i.e., documents may be relevant and irrelevant to the query). Accurate ordering or ranking of these documents according to their relevance is of paramount importance to effective search. Many features may affect the relevance of documents, and thus, it is difficult to adapt ranking functions manually. A classifier can be used as an alternate approach to approximate the relevance of these documents. In this case, according to [23], class membership probabilities (i.e., $\hat{p}(\text{relevant}|x), \ldots, \hat{p}(\text{irrelevant}|x)$) can be directly used to approximate the relevance of document x.

For these examples, it is important to estimate class membership probabilities, $\hat{p}(c_1|x),\ \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$, as reliably as possible, for each instance x. A calibrated classifier[1] is one which provides reliable estimates of class membership. Intuitively, if $\hat{p}(c_i|x) \approx p(c_i|\hat{p}(c_i|x))$ for all x, then the classifier is said to be well calibrated.[2] Accurate classifiers are not necessarily well calibrated. Maximum margin classifiers, such as SVMs, push probability mass away from 0 and 1, yielding a characteristic sigmoid shaped distortion in the estimated membership probabilities [30]. Naive Bayes classifiers, which make unrealistic independence assumptions, push probability estimates closer to 0 and 1 [29]. Boosting classifiers can be viewed as an additive logistic regression model [10], and thus, the predictions performed by these classifiers fit a logit of the true probabilities, as opposed to the true probabilities themselves. Decision Tree classifiers try to make leaves homogeneous, and, thus, the estimates are systematically moved closer to 1 [18]. In general, classifiers that follow an accuracy maximization strategy are not well calibrated.

The proposed associative classifiers, on the other hand, do not follow any accuracy maximization strategy, neither make unrealistic assumptions. Instead, they simply use the extracted rules to describe the training data. This description is then used to estimate class membership probabilities. If the description is accurate, then the estimates are likely to be close to the actual probabilities. However, producing an accurate description of the training data is challenging. This is mainly because practical limitations impose the need of frequency-based pruning methods to avoid rule explosion, and, thus important rules may not be included in the description, hurting calibration. An alternate approach to avoid rule explosion is to extract rules on a demand-driven basis, according to the instance being classified. This approach is known as *lazy associative classification* (LAC), and it has been shown that important rules are more likely to be included in the description of the training data [25]. To further improve the effectiveness of lazy associative classifiers, some mechanisms can be used to calibrate the estimates.

Calibration mechanisms have already been studied in weather forecasting, game theory, in the field of artificial intelligence to parameter estimation of evolutionary approaches such as genetic algorithms [11], and recently in data mining techniques such as clustering [12] and classification [3]. Most of the classifiers demand sophisticated calibration mechanisms, such as Platt Scaling [20] or Isotonic Regression [30] (for SVMs), Logistic Correction [10] (for boosting-based classifiers), m-estimation [2] (for Naive-Bayes classifiers), and Curtailment [29] (for Decision Tree classifiers). In addition to classification, other data mining technique as clustering can be improved using calibration. Moreover, other machine learning techniques. In this paper, we propose to calibrate associative classifiers using an entropy minimization mechanism. The key insight is to divide the probability space into bins, in a way that the entropy of each bin is as minimum as possible (i.e., each bin contains probabilities associated with predictions that are either correct or incorrect). To evaluate the effectiveness of the

[1] The notion of calibration (sometimes termed reliability), was introduced in [6] to describe situations where the observed frequencies of events match the probabilities forecasted for them.

[2] As an illustration, consider all the examples for which a classifier assigns a probability $\hat{p}(c_i|x) = 0.90$. If this classifier is calibrated, then nearly 90% of these examples are correctly classified (i.e., the empirical probability, $p(c|\hat{p}(c|x))$, is close to 0.90). In this case, $p(c_i|\hat{p}(c_i|x)) \approx 0.90 \approx \hat{p}(c_i|x)$.

calibrated classifiers, we performed a systematic set of experiments using datasets obtained from real-world and challenging applications. Our results suggest that calibrated lazy associative classifiers achieve better performance than SVM, Naive Bayes, and Decision Tree classifiers (even after calibrating these classifiers) in applications where calibration is necessary. The specific contributions of this paper are:

- We propose novel calibration mechanisms. These mechanisms employ a histogram technique, known as binning, which divides the probability space into bins. Bin boundaries are set in a way that minimizes entropy, so that each bin contains probabilities that are either associated with correct or incorrect predictions. In contrast to previous work [24], our proposed mechanism automatically sets the number of bins using the MDL principle. We evaluated such calibration mechanisms using datasets obtained from complex real-world applications.
- We show that classifiers that maximize accuracy are usually non-calibrated and ill-suited for cost-sensitive and cautious classification problems. We also show that LAC, which simply describes the training data, is more propitious to these classification problems.
- We show that calibrated LAC achieves superior performance than other classifiers, such as SVMs, Naive Bayes and Decision Trees, even after these classifiers are calibrated using specific calibration mechanisms [10,20,30].
- We show that calibrated LAC is able to place test instances into two distinct zones – a "safe zone" (where predictions are very likely to be correct) and a "danger zone" (where predictions are doubtful). This ability enables this classifier to (i) use reliable predictions for (self) training, and (ii) abstain from doubtful predictions.

Preliminary results concerning the evaluation of calibrated lazy associative classification were published in [24]. Major extensions are presented in this paper. First, the calibration mechanism proposed in [24] is parameter-laden, requiring the setting of input parameters in order to calculate the number of bins and the corresponding boundaries. In this paper, a parameter-free, MDL-based calibration mechanism is proposed. Second, we evaluate the effectiveness of the proposed MDL-based entropy minimization calibration mechanism when coupled with classifiers, others than LAC. Finally, we present novel application scenarios for which calibrated classification is important.

2. Related work

There are several studies investigating calibration of classifiers, including SVMs, Naive Bayes, and Decision Trees. The calibration of Naive Bayes and Decision Tree classifiers were investigated in [29], where it was shown that probabilities estimated by these classifiers are usually far from the observed, true probabilities. The same methodology was used to show that SVM classifiers are poorly calibrated [30], and that the distortion very often forms a sigmoid pattern. Boosting-based classifiers were shown to be poorly calibrated in [18]. Although these classifiers are not calibrated, they tend to assign higher probabilities to the correct class. Therefore, these classifiers are usually accurate. The advantages of calibrated classifiers were discussed in [3], where it was shown that calibrating a classifier is guaranteed not to decrease its accuracy.

There are several existing mechanisms used to correct the distortion between estimated and observed probabilities. Mechanisms for calibrating SVM classifiers transform the predictions to posterior probabilities by passing them through a sigmoid (logistic regression). The parametric approach proposed in [20] consists in finding the parameters a and b for a sigmoid function of the form $\hat{p}_c(c_i|x) = \frac{1}{(1+\exp\{a\hat{p}(c_i|x)+b\})}$, which transforms the original estimated probability, $\hat{p}(c_i|x)$, into a calibrated estimate, $\hat{p}_c(c_i|x)$. Boosting-based classifiers are also calibrated using a mechanism which is based on logistic regression [10]. This mechanism transforms original estimates, $\hat{p}(c_i|x)$, into calibrated estimates, $\hat{p}_c(c_i|x)$, using the function $\hat{p}_c(c_i|x) = \frac{1}{(1+\exp\{-2a\hat{p}(c_i|x)\})}$. These parameters are found by minimizing the negative log-likelihood of the data. Mechanisms for calibrating Decision Tree and Naive Bayes classifiers were proposed in [29]. These mechanisms are based on smoothing the distribution of the original estimates, and they also rely on finding parameters from the data. In [30], a calibration mechanism based on binning the probability space was proposed. The mechanism is based on a technique known as Isotonic Regression [15,19], in which the number of bins is automatically adjusted by enforcing monotonicity. All the mentioned mechanisms find the corresponding parameters through 10-fold cross-validation using the training data.

3. Associative classification

Associative classification exploits the fact that, frequently, there are strong associations between features $(a_1, a_2, \ldots, a_m)$ and classes $(c_1, c_2, \ldots, c_n)$. The learning strategy adopted by associative classifiers is based on uncovering such associations from the training data (denoted as $\mathcal{D}$), and then building a function$\{a_1, a_2, \ldots, a_m\} \rightarrow \{c_1, c_2, \ldots, c_n\}$ using such associations. Typically, these associations are expressed using rules of the form $\mathcal{X} \rightarrow c_1, \ldots, \mathcal{X} \rightarrow c_n$, where $\mathcal{X} \subseteq \{a_1, \ldots, a_m\}$. In the following discussion we will denote as $\mathcal{R}$ an arbitrary rule set. Similarly, we will denote as $\mathcal{R}_{c_i}$ a subset of $\mathcal{R}$ which is composed of rules of the form $\mathcal{X} \rightarrow c_i$. A rule $\mathcal{X} \rightarrow c_i$ is said to match instance x if $\mathcal{X} \subseteq x$ (x contains all features in $\mathcal{X}$), and these rules form the rule set $\mathcal{R}^x_{c_i}$. That is, $\mathcal{R}^x_{c_i}$ is composed of rules predicting class c_i and matching instance x. Obviously, $\mathcal{R}^x_{c_i} \subseteq \mathcal{R}_{c_i} \subseteq \mathcal{R}$.

Naturally, there is a total ordering among rules, in the sense that some rules show stronger associations than others. Two widely used statistic measures, called support (denoted as $\sigma(\mathcal{X} \rightarrow c_i)$) and confidence (denoted as $\theta(\mathcal{X} \rightarrow c_i)$), express the

strength of the association between $\mathcal{X}$ and c_i. The absolute support of a rule $\mathcal{X} \rightarrow c_i$ is simply given by the number of examples in the training data, $\mathcal{D}$, belonging to class c_i and having $\mathcal{X}$ as a subset, as shown in Eq. (1).

$$\pi(\mathcal{X} \rightarrow c_i) = |x \in \mathcal{D} \text{ such that } \mathcal{X} \subseteq x \text{ and } c_i \text{ is the class for } x| \tag{1}$$

The support of a rule $\mathcal{X} \rightarrow c_i$ is the fraction of examples in $\mathcal{D}$, belonging to class c_i and having $\mathcal{X}$ as a subset, as shown in Eq. (2).

$$\sigma(\mathcal{X} \rightarrow c_i) = \frac{\pi(\mathcal{X} \rightarrow c_i)}{|\mathcal{D}|} \tag{2}$$

The confidence of a rule $\mathcal{X} \rightarrow c_i$ is the conditional probability of c_i being the correct class for an example x, given that $\mathcal{X} \subseteq x$, as shown in Eq. (3).

$$\theta(\mathcal{X} \rightarrow c_i) = \frac{\pi(\mathcal{X} \rightarrow c_i)}{|x \in \mathcal{D} \text{ such that } \mathcal{X} \subseteq x|} \tag{3}$$

The probability (or likelihood) of instance x being member of class c_i is estimated by combining rules in $\mathcal{R}^x_{c_i}$. An effective strategy is to interpret $\mathcal{R}^x$ as a poll, in which rule $\mathcal{X} \rightarrow c_i \in \mathcal{R}^x$ is a vote given by $\mathcal{X}$ for class c_i. The weight of a vote $\mathcal{X} \rightarrow c_i$ depends on the strength of the association between $\mathcal{X}$ and c_i, which is $\theta(\mathcal{X} \rightarrow c_i)$. The process of estimating the probability of x being member of c_i starts by summing weighted votes for class c_i and then averaging the obtained value by the total number of votes for c_i, as expressed by the score $s(c_i,x)$ shown in Eq. (4) (where $r_j \subseteq \mathcal{R}^x_{c_i}$ and $|\mathcal{R}^x_{c_i}|$ is the cardinality of $\mathcal{R}^x_{c_i}$). Thus, the score $s(c_i,x)$ gives the average confidence of the rules in $\mathcal{R}^x_{c_i}$. The estimated probability of x being member of c_i, $\hat{p}(c_i|x)$, is simply obtained by normalizing $s(c_i,x)$, as shown in Eq. (5). A higher value of $\hat{p}(c_i,x)$ indicates a higher likelihood of x being member of c_i. The class associated with the highest likelihood is finally predicted.

$$s(c_i,x) = \frac{\sum_{j=1}^{|\mathcal{R}^x_{c_i}|} \theta(r_j)}{|\mathcal{R}^x_{c_i}|} \tag{4}$$

$$\hat{p}(c_i|x) = \frac{s(c_i,x)}{\sum_{j=1}^{n} s(c_j,x)} \tag{5}$$

3.1. Lazy associative classification (LAC)

Rule extraction is a major issue when devising an associative classifier. Extracting all rules is frequently unfeasible, and, thus, pruning strategies are employed in order to reduce the number of rules that are processed. The typical pruning strategy is based on a minimum support threshold, $\sigma_{\min}$, which separates frequent from infrequent rules. Ideally, infrequent rules are not important. However, it is often the case that infrequent rules indicate true *feature-class* associations, and are, therefore, important for sake of classification. An optimal minimum support threshold is unlikely to exist, and tuning is generally driven by intuition, and prone to error as a consequence.

Typically, a $\sigma_{\min}$ value is employed and a single rule set $\mathcal{R}$ is extracted from the training data, $\mathcal{D}$. This rule set is a description of $\mathcal{D}$ and is used to classify all test instances. Two problems may arise: (i) a not so frequent but important rule may not be extracted from $\mathcal{D}$ (hurting the quality of the description), and (ii) a useless[3] rule can be extracted from $\mathcal{D}$ (incurring unnecessary overhead). An ideal scenario would be to extract only useful rules from $\mathcal{D}$, without discarding important ones. Test instances have valuable information that can be used during rule extraction to guide the search for useful and important rules. We propose to achieve the ideal scenario by extracting rules on a demand-driven basis, according to the test instance being considered. Specifically, whenever a test instance t is being considered for classification, that instance is used as a filter to remove irrelevant features (and often entire examples) from $\mathcal{D}$, forming a *projected training data*, $\mathcal{D}^t$, which contains only features that are included in instance t [25]. This process reduces the size and dimensionality of the training data, since irrelevant features and examples are not considered while extracting the rules for a specific test instance.

A typical strategy used to prevent support-based over-pruning (i.e., discarding important rules) is to use a different cut-off value, which is calculated depending on the frequency of the classes. More specifically, the cut-off value is higher for rules predicting more frequent classes, and lower for rules predicting less frequent classes. The problem with this strategy is that it does not take into account the frequency of the features composing the rule, and, thus, if an important rule is composed of rare features, it will be discarded, specially if this rule predicts a highly frequent class. We propose an alternate strategy that employs multiple cut-off values, which are calculated depending on the frequency of the features composing a test instance. Intuitively, if a test instance t contains frequent features (i.e., these features occur in many examples in $\mathcal{D}$), then the size of the projected training data will be large. Otherwise, if a test instance t contains rare features (i.e., these features occur only in few examples in $\mathcal{D}$), then the size of the projected training data will be small. For a fixed value of $\sigma_{\min}$, the cut-off value for instance t, π^t, is calculated based on the size of the corresponding projected training data, that is, $\pi^t = \sigma_{\min} \times |\mathcal{D}^t|$. The cut-off value applied while considering test instance t varies from $1 \leqslant \pi^t \leqslant \sigma_{\min} \times |\mathcal{D}|$, which is bounded by $\sigma_{\min} \times |\mathcal{D}|$ (the single

[3] A rule is useless if it does not match any test instance in $\mathcal{T}$.

cut-off value applied by the typical support-based pruning strategy). Therefore, the chance of discarding important (but less frequent) rules is reduced. The main steps of lazy associative classification are shown in Algorithm 1.

Algorithm 1. Lazy associative classification.

Require: Examples in $\mathcal{D}$, $\sigma_{\min}$, and test instance t
Ensure: The predicted class for instance t
1: Let $\mathcal{L}(a_i)$ be the set of examples in $\mathcal{D}$ in which feature a_i occurs
2: $\mathcal{D}^t \Leftarrow \emptyset$
3: **for each** feature $a_i \in t$ **do** $\mathcal{D}^t \Leftarrow \mathcal{D}^t \cup \mathcal{L}(a_i)$
4: $\pi^t \Leftarrow \sigma_{\min} \times |\mathcal{D}^t|$
5: **for each** c_i **do** $\mathcal{R}^t_{c_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_i$, such that $\mathcal{X} \subseteq t$ and $\pi(\mathcal{X} \rightarrow c_i) \geqslant \pi^t$
6: Estimate $\hat{p}(c_1|x), \ldots, \hat{p}(c_n|x)$, and predict the class with the highest probability

Different test instances may demand different rule sets, but different rule sets often share common rules. In this case, caching is very effective in reducing work replication. An efficient caching approach was proposed in [23], and is also used here.

4. Calibration of associative classifiers

In this section we define calibrated classifiers, and then we propose mechanisms to calibrate the associative classifier described in Section 3. We finalize this section by discussing the advantages of well calibrated classifiers.

4.1. ϵ-Calibrated classifier

The calibration of a classifier can be visualized using reliability diagrams. Diagrams for two arbitrary classifiers on an arbitrary dataset are depicted in Fig. 1 (left). These diagrams are built as follows [7]. First, the probability space (i.e., the x-axis) is divided into a number of bins, which was chosen to be 10 in our case. Probability estimates (i.e., theoretical probabilities) with value between 0 and 0.1 fall in the first bin, estimates with value between 0.1 and 0.2 fall in the second bin, and so on. The fraction of correct predictions associated with each bin, *which is the true, empirical probability* (i.e., $p(c|\hat{p}(c|x))$), is plotted against the estimated probability (i.e., $\hat{p}(c|x)$). If the classifier is well calibrated, the points will fall near the diagonal line, indicating that estimated probabilities are close to empirical probabilities.

The degree of calibration of a classifier, denoted as ϵ, may be obtained by simply measuring the discrepancy between observed probabilities (o_i) and estimated probabilities (e_i), as shown in Eq. (6) (where n is the number of bins). Values of ϵ range from 0 to 1. A value of 0 means that there is no relationship between estimated probabilities and true probabilities. A value of 1 means that all points lie exactly on a straight line with no scatter. Classifier 1 is better calibrated than Classifier 2, as shown in Fig. 1 (right).

$$\epsilon = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{(o_i - e_i)^2}{(o_i + e_i)^2} \tag{6}$$

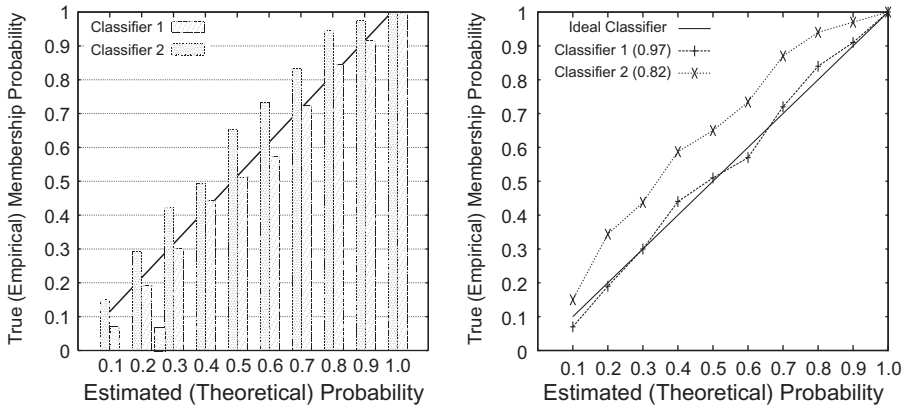

Fig. 1. Left – reliability diagram. Right – classifiers with $\epsilon = 0.97$ and $\epsilon = 0.82$.

4.2. Calibration based on entropy minimization

To transform original probability estimates, $\hat{p}(c_i|x)$, into accurate well calibrated probabilities, we also use a method based on binning. The method starts by estimating membership probabilities using the training data. A typical strategy is 10-fold cross-validation. In this case, the training data, $\mathcal{D}$, is divided into 10 partitions, and at each trial, 9 partitions are used for training the classifier, while the remaining partition is used to simulate a test set. After the 10 trials, the classifier will have stored in the set $\mathcal{O}$, the membership probability estimates for all examples in $\mathcal{D}$. This process is shown in Algorithm 2.

Algorithm 2. Estimating membership probabilities.

Require: Examples in $\mathcal{D}$
Ensure: For each example x in $\mathcal{D}$, the corresponding membership probabilities $\hat{p}(c_1|x), \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$, along with the correct class

1: $\mathcal{O} \Leftarrow \emptyset$
2: Split $\mathcal{D}$ into 10 equal-sized partitions, $d_1, d_2, \ldots, d_{10}$
3: **for each** partition d_i **do**
4: **for each** example $x \in d_i$ **do**
5: Estimate probabilities, $\hat{p}(c_1|x), \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$, using $\{\mathcal{D}\text{-}d_i\}$ as training
6: $\mathcal{O} \Leftarrow \mathcal{O} \cup \{(\hat{p}(c_1|x), v_1)\} \cup \ldots \cup \{(\hat{p}(c_n|x), v_n)\}$, where $v_i = 1$ if c_i is the correct class for example x, and $v_i = 0$ otherwise
7: **end for**
8: **end for**
9: **return** $\mathcal{O}$

Once the probabilities are estimated, a Naive strategy would proceed by first sorting these probabilities in ascending order (i.e., the probability space), and then dividing them into k equal-sized bins, each having pre-specified boundaries. An estimate is placed in a bin according to its value (i.e., values between 0 and $\frac{1}{k}$ are placed in the first bin, values between $\frac{1}{k}$ and $\frac{2}{k}$ in the second, and so on). The probability associated with a bin is given by the fraction of correct predictions that were placed in it. An estimate $\hat{p}(c_i|x)$ is finally calibrated by using the probability associated with the corresponding bin. Specifically, each bin $b_{l\hookrightarrow u} \in \mathcal{B}$ (with l and u being its boundaries) works as a map, relating estimates $\hat{p}(c_i|x)$ (such that $l \leqslant \hat{p}(c_i|x) < u$) to the corresponding calibrated estimates, $p_{b_{l\hookrightarrow u}}$. Thus, this process essentially discretizes the probability space into intervals, so that the accuracy associated with the predictions in each interval is as reliable as possible.

Such Naive strategy, however, can be disastrous as critical information may be lost due to inappropriate bin boundaries. Instead, we propose to use information entropy associated with candidate bins to select the boundaries [8]. To illustrate the method, suppose we are given a set of pairs $(\hat{p}(c_i|x), v) \in \mathcal{O}$, where v can take the values 0 (the prediction is wrong) or 1 (otherwise), as shown in step 6 of Algorithm 2. In this case, the entropy of $\mathcal{O}$ is given by Eq. (7).

$$E(\mathcal{O}) = -\frac{|(\hat{p}(c_i|x), 0) \in \mathcal{O}|}{|\mathcal{O}|} \times \log\left(\frac{|(\hat{p}(c_i|x), 0) \in \mathcal{O}|}{|\mathcal{O}|}\right) - \frac{|(\hat{p}(c_i|x), 1) \in \mathcal{O}|}{|\mathcal{O}|} \times \log\left(\frac{|(\hat{p}(c_i|x), 1) \in \mathcal{O}|}{|\mathcal{O}|}\right) \tag{7}$$

There is a threshold f, which is a boundary that induces two partitions of $\mathcal{O}$ ($b_{f\leqslant}$ and $b_{f>}$, where $b_{f\leqslant}$ contains pairs $(\hat{p}(c_i|x), v)$ for which $\hat{p}(c_i|x) \leqslant f$, and $b_{f>}$ contains pairs for which $\hat{p}(c_i|x) > f$). According to Fayyad and Irani in [8], the selected threshold, t, is the one which minimizes the weighted average entropies, given by Eq. (8).

$$E(\mathcal{O}, f) = \frac{|b_{f\leqslant}|}{|\mathcal{O}|} \times E(b_{f\leqslant}) + \frac{|b_{f>}|}{|\mathcal{O}|} \times E(b_{f>}) \tag{8}$$

This method is then applied recursively to both of the partitions induced by t, $b_{t\leqslant}$ and $b_{t>}$, creating multiple intervals until a stopping criterion is fulfilled. Following the procedure proposed in [8], splitting stops if the information gain (the difference between the entropies before and after the split) is lower than the minimum description length [22] of the partition, and the final set of bins, $\mathcal{B}$, is found. Using Eq. (8), the information gain obtained by applying a threshold t over a partition $\mathcal{O}$, is given in Eq. (9) (as discussed in [8]).

$$E(\mathcal{O}) - E(\mathcal{O}, t) = E(\mathcal{O}) - \frac{|b_{t\leqslant}|}{|\mathcal{O}|} \times E(b_{t\leqslant}) - \frac{|b_{t>}|}{|\mathcal{O}|} \times E(b_{t>}) \tag{9}$$

Also, according to [8], the minimum description length induced by a threshold t over a partition $\mathcal{O}$ is the second term of the following inequality:

$$E(\mathcal{O}) - E(\mathcal{O}, t) > \frac{\log(|\mathcal{O}| - 1)}{|\mathcal{O}|} + \frac{\Delta(\mathcal{O}, t)}{|\mathcal{O}|} \tag{10}$$

Plugging Eq. (9) into Eq. (10) (please, notice that Eqs. (8)–(10) have first appeared in [8]), we obtain $\Delta(\mathcal{O}, t) = \log(3^k - 2) - (k \times E(\mathcal{O}) - k_1 \times E(b_{t\leqslant}) - k_2 \times E(b_{t>}))$, where k_i is either 1 or 2 ($k_i = 1$ if the corresponding partition is pure, that is, if it contains only correct (or only incorrect) predictions; and $k_i = 2$ otherwise).

Finally, for each instance x in the test set, the corresponding probabilities $\hat{p}(c_1|x), \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$ are estimated using the entire training data, $\mathcal{D}$. Then, the estimated probabilities are calibrated using the accuracy associated with the appropriate bin in $\mathcal{B}$ (which is the fraction of correctly classified instances in each bin), as shown in Algorithm 3.

Algorithm 3. Outputting calibrated probability estimates.

Require: Examples in $\mathcal{D}$, instances in $\mathcal{T}$, the calibrated probability $p_{b_{l\hookrightarrow u}}$ of each bin $b_{l\hookrightarrow u}$
Ensure: For each estimate $\hat{p}(c_i|x)$, the corresponding calibrated estimate $\hat{p}_c(c_i|x)$

1: **for each** instance $x \in \mathcal{T}$
2: Estimate probabilities, $\hat{p}(c_1|x), \hat{p}(c_2|x), \ldots, \hat{p}(c_n|x)$, using $\mathcal{D}$ as training
3: **for each** c_i **do output** $\hat{p}_c(c_i|x) = p_{b_{l\hookrightarrow u}}$, such that $l \leqslant \hat{p}(c_i|x) < u$
4: **end for**

4.2.1. Illustrative example

Table 1 shows an illustrative example composed of 10 documents extracted from a digital library. Each document belongs to one category. Such documents are given as training data. Several rules are extracted from these documents. Specifically, $\mathcal{R}^{d_1}$, which is the set of rules (extracted from $\{\mathcal{D} - d_1\}$) matching document d_1, includes:

1. text = system (s) → Inf. Retrieval (θ = 1.00)
2. text = {rule (s) ∧ database (s)} → Data Mining (θ = 1.00)
3. text = rule (s) → Data Mining (θ = 1.00)
4. text = database (s) → Databases (θ = 0.40)
5. text = database (s) → Data Mining (θ = 0.40)
6. text = database (s) → Inf. Retrieval (θ = 0.20)

From such decision rules, class membership probabilities for document d_1 are estimated using Eq. (5), resulting in the following probabilities: $\hat{p}(\text{Databases}|d_1) = 0.19, \hat{p}(\text{Data Mining}|d_1) = 0.44$, and $\hat{p}(\text{Inf. Retrieval}|d_1) = 0.37$. Membership probabilities for all documents are shown in Table 2.

Fig. 2 (left) shows the process of setting bin boundaries by entropy minimization, for category "Data Mining". Initially, the probability space (which ranges from 0.00 to 1.00) is divided into two bins. The cut point at 0.45 gives an information gain which is higher than the minimum description length of initial bin. Now, there are two bins. The bin on the right (i.e., [0.45–1.00]) is not divided anymore, since additional cut points would not provide enough information gain. The bin on the left is further divided into two bins. The cut point at 0.20 gives an information gain which is higher than the minimum description length of this bin. Then, the process stops because no more bins are created. Fig. 2 (right) shows the same process of setting bin boundaries for category "Inf. Retrieval".

Bins obtained for each category are shown in Table 3. For this simplified example, only two bins are produced for category "Databases", and only three bins are produced for categories "Data Mining" and "Inf. Retrieval". The calibrated probability for each bin, which is the fraction of correct predictions within each bin, are also shown in Table 3.

4.2.2. Safe zone and danger zone

Test instances may be divided into two zones. The safe zone contains instances for which the corresponding predictions are likely to be correct. These test instances can be used to enhance the training data. More specifically, the training data is augmented with a new example, which is composed by the features of the test instance along with its corresponding prediction (which is likely to be correct). Since LAC generates rules at classification time, the next instances to be classified can take advantage of information coming from instances in the safe zone that were inserted in the training data. The danger zone, on the other hand, contains instances for which the predictions are doubtful, and the classifier can abstain from predicting the class of such instances. A threshold $r_{\min}$ (i.e., a minimum reliability) delimits these two zones.

Table 1
Example using documents of a digital library. Features are words in the document title.

	Category	Features
d_1	Databases	Rules in database systems
d_2	Databases	Applications of logic databases
d_3	Databases	Hypertext databases and data mining
d_4	Data Mining	Mining association rules in large databases
d_5	Data Mining	Database mining: a performance perspective
d_6	Data Mining	Algorithms for mining association rules
d_7	Inf. Retrieval	Text databases and information retrieval
d_8	Inf. Retrieval	Information filtering and information retrieval
d_9	Inf. Retrieval	Term weighting approaches in text retrieval
d_{10}	Inf. Retrieval	Performance of information retrieval systems

Table 2
Class membership probabilities. The number between parenthesis indicates if the prediction is correct (1) or not (0).

	$\hat{p}$(Databases|d)	$\hat{p}$(Data Mining|d)	$\hat{p}$(Inf. Retrieval|d)
d_1	0.19 (1)	0.44 (0)	0.37 (0)
d_2	0.27 (1)	0.48 (0)	0.24 (0)
d_3	0.26 (1)	0.61 (0)	0.13 (0)
d_4	0.36 (0)	0.46 (1)	0.18 (0)
d_5	0.27 (0)	0.25 (1)	0.48 (0)
d_6	0.30 (0)	0.53 (1)	0.17 (0)
d_7	0.22 (0)	0.25 (0)	0.53 (1)
d_8	0.00 (0)	0.29 (0)	0.71 (1)
d_9	0.00 (0)	0.00 (0)	1.00 (1)
d_{10}	0.31 (0)	0.31 (0)	0.38 (1)

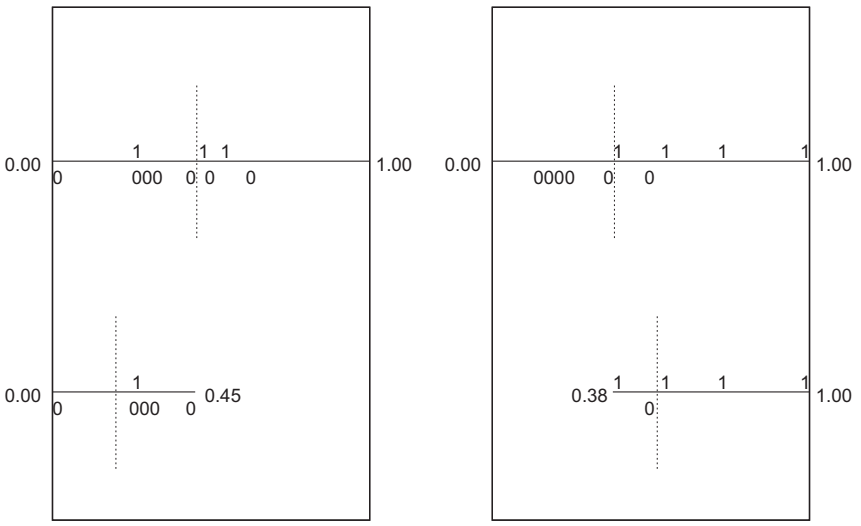

Fig. 2. Calculating bin boundaries. Left – category "Data Mining". Right – category "Inf. Retrieval".

Table 3
Bin boundaries and calibrated probabilities for each category.

Databases		Data Mining		Inf. Retrieval	
Boundaries	Prob.	Boundaries	Prob.	Boundaries	Prob.
[0.00–0.17]	0.000	[0.00–0.20]	0.000	[0.00–0.38]	0.000
[0.17–1.00]	0.375	[0.20–0.45]	0.200	[0.38–0.52]	0.500
		[0.45–1.00]	0.500	[0.52–1.00]	1.000

4.2.3. Estimating the accuracy after a prediction

In several applications it is desirable to have a reliable estimate of classification accuracy. Well calibrated classifiers can estimate the accuracy after each prediction by simply summing the calibrated probability associated with the prediction. If the classifier is well calibrated the estimated accuracy will be close to the actual accuracy.

5. Evaluation

In this section we present the experimental results for the evaluation of the proposed calibrated lazy associative classifiers, hereafter referred to as CaLAC (Naive), CaLAC (MDL) and CaLAC (MDL + SZ). CaLAC (Naive) is the classifier obtained after calibrating LAC using the Naive Binning mechanism. Similarly, CaLAC (MDL) is the classifier obtained after calibrating LAC using the entropy minimization mechanism, and CaLAC (MDL + SZ) is the classifier obtained using the entropy minimization mechanism, and, in this case, instances in the safe zone are incorporated into the training data, as discussed in Section 4.2.2.

Our evaluation is based on a comparison against current state-of-the-art classifiers, which include SVM [14], Naive Bayes [5], and Decision Tree classifiers [21]. After being calibrated using specific mechanisms [20,2,29], these classifiers are, respectively, referred to as CaSVM, CaNB, and CaDT. For CaLAC (Naive), CaLAC (MDL) and CaLAC (MDL + SZ), we set $\sigma_{\min} = 0.001$.

For CaLAC (MDL + SZ) we set $r_{min} = 0.90$. For CaSVM we used linear kernels and set the regularization parameter $C = 0.90$ in the first application, $C = 2.00$ in the second application, and $C = 0.01$ in the third application. These parameters were set using the LIBSVM tool for parameter search (available at http://www.csie.ntu.edu.tw/cjlin/libsvmtools/#13). All parameters (for CaLAC and CaSVM) were obtained using a separated validation set. For CaNB and CaDT we used the default parameters, which were also used in other works [5].

All experiments were run on an 1.8 MHz Intel processors with 2 GB main memory running Linux. Instead of performing experiments on many datasets that are relatively small and easy to achieve good performance on, we prefer to do a detailed investigation using complex datasets obtained from three real-world applications that are known to be challenging.

5.1. Cautious classification – the ACM digital library

For this application we used a dataset called ACM-DL, which was extracted from the ACM Computing Classification System (http://portal.acm.org/dl.cfm/). ACM-DL is a set of 6682 documents (metadata records) labeled using the 8 first level categories of ACM (General Literature, Hardware, Computer Systems Organization, Software, Data, Theory of Computation, Mathematics of Computing, Information Systems, Computing Methodologies, Computer Applications, Computing Milieux). Features in ACM-DL include words (in title and abstracts) and citations to other documents. ACM-DL has a vocabulary of 9840 unique words, and a total of 11,510 internal citations and 40,387 citations for documents outside of the digital library. The classifier must decide to which category a document belongs. However, the administrator of the digital library imposes an additional minimum accuracy requirement, acc_{min}, to the classifier. In this case, the classifier must estimate the total accuracy after each prediction is performed, and then it must decide to continue classifying documents (if the estimated accuracy is higher than acc_{min}) or to stop classification (if the estimated accuracy is lower than acc_{min}). We performed 10-fold cross-validation and the final result represents the average of the 10 runs (all results to be presented were found statistically significant at the 95% confidence level when tested with a two-tailed paired t-test).

5.1.1. Calibrating the classifiers

We employed the two mechanisms described in Section 4.2 to calibrate LAC – the Naive Binning mechanism and the MDL-based entropy minimization (MDL) mechanism. The number of bins for the Naive Binning mechanism was set to 5. The bins obtained for the "Information Systems" category are shown in Fig. 3 (left). Bin boundaries (i.e., original probability estimates) are shown in the x-axis and the corresponding calibrated probabilities are shown in the y-axis. Coincidentally, for the "Information Systems" category, the MDL-based entropy minimization mechanism produced 5 bins of varying sizes, which are shown in Fig. 3 (right).

After the bins are found, we apply LAC to the test set, and we replace the original probability estimate (x-axis) by the calibrated probability associated with the corresponding bin (y-axis). The result of calibration is depicted in Fig. 4, which shows ϵ values for LAC, before and after being calibrated. Other classifiers were also evaluated. The worst classifier in terms of calibration, is SVM with $\epsilon = 0.69$. After calibrating SVM, the corresponding classifier CaSVM shows $\epsilon = 0.75$. NB and LAC, with $\epsilon = 0.76$ and $\epsilon = 0.78$, respectively, are already better calibrated than CaSVM. These classifiers, when calibrated, show the best calibration degrees-CaNB with a $\epsilon = 0.91$, and CaLAC (MDL) with $\epsilon = 0.97$.

Table 4 shows the degree of calibration achieved by SVM, DT, and NB, using several calibration mechanisms, and the proposed mechanism based on entropy minimization. As it can be seen, Platt Scaling [20], which is the specific mechanism for

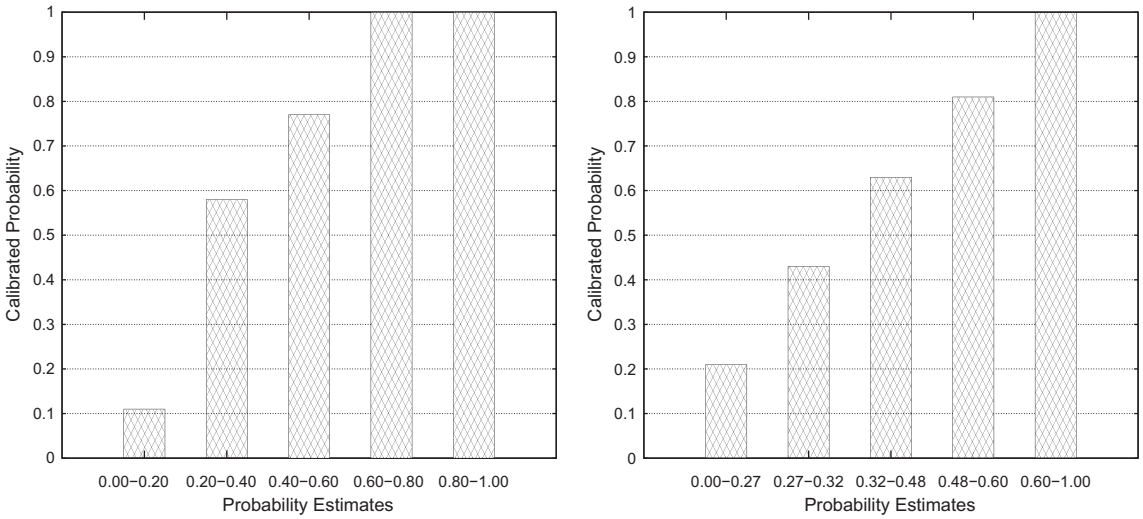

Fig. 3. Information systems category. Left – Naive Binning mechanism. Right – MDL-based entropy minimization mechanism.

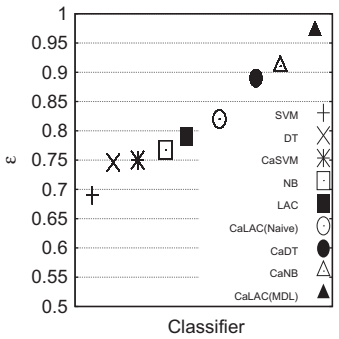

Fig. 4. Classifiers, before, and after calibration using specific mechanisms.

Table 4
Comparing different calibration mechanisms.

Classifier	Calibration	ϵ
SVM	Platt Scaling	0.75
	Isotonic Regression	0.73
	MDL-based	0.69
DT	Isotonic Regression	0.87
	Curtailment	0.89
	MDL-based	0.92
NB	Binning ($n = 10$)	0.91
	m-Estimation	0.92
	Isotonic Regression	0.92
	MDL-based	0.94
LAC	Naive	0.82
	Isotonic Regression	0.92
	MDL-based	0.97

calibrating SVMs, achieves higher calibration degrees. For SVMs, the proposed entropy minimization mechanism produced several bins, and most of these bins contain very few samples. Such small bins may provide inaccurate probability estimates. On the other hand, the proposed MDL-based calibration mechanism provides higher calibration degrees than Curtailment [29] (for DTs), simple binning [2] (for NB), and Isotonic Regression [30,19] (for LAC).

5.1.2. Results

We start our analysis by evaluating each classifier in terms of its ability for estimating the actual accuracy. Fig. 5 shows the actual accuracy and the accuracy estimates for each classifier, so that the corresponding values can be directly compared.[4] As expected, CaLAC (MDL) and CaLAC (MDL + SZ) show to be better calibrated than CaLAC (Naive). This is because the bins used by CaLAC (Naive) are produced in an ad hoc way, while the bins used by CaLAC (MDL) and CaLAC (MDL + SZ) are produced following the entropy minimization strategy. The direct consequence of such strategy is that a bin is likely to contain predictions which are as similar as possible (i.e., a bin is likely to contain either correct or incorrect predictions). While in most of the cases CaNB and CaDT are well calibrated, CaSVM very often underestimates or overestimates the actual accuracy, and is poorly calibrated. This result is supported by the results presented in [3] (which show that Naive Bayes classifiers are much better calibrated than SVM classifiers).

If the administrator of the digital library specifies a threshold $acc_{\min}$ (i.e., the minimum acceptable accuracy of the classifier), then the value of the classifier resides in how many documents it is able to classify while respecting $acc_{\min}$. Fig. 6 shows the fraction of documents in the test set each classifier is able to classify for a given $acc_{\min}$. Clearly, CaLAC (MDL) and CaLAC (MDL + SZ) are the best performers, except for $acc_{\min}$ values higher than 0.95, when the best performer is CaLAC (Naive). CaNB and CaDT are in close rivalry, with CaDT being slightly superior. In most cases, both CaNB and CaDT are superior to CaLAC (Naive). CaSVM, as expected, is the worst performer for all values of $acc_{\min}$.

5.2. Cost-sensitive classification – KDDCUP'98

For this application we used a dataset called KDD-98, which was used in KDDCUP'98 contest. This dataset was provided by the Paralyzed Veterans of America (PVA), an organization which devises programs and services for US veterans. With a database of over 13 million donors, PVA is also one of the world's largest direct mail fund raisers.

[4] For each experiment, predictions were sorted from the most reliable to the least reliable.

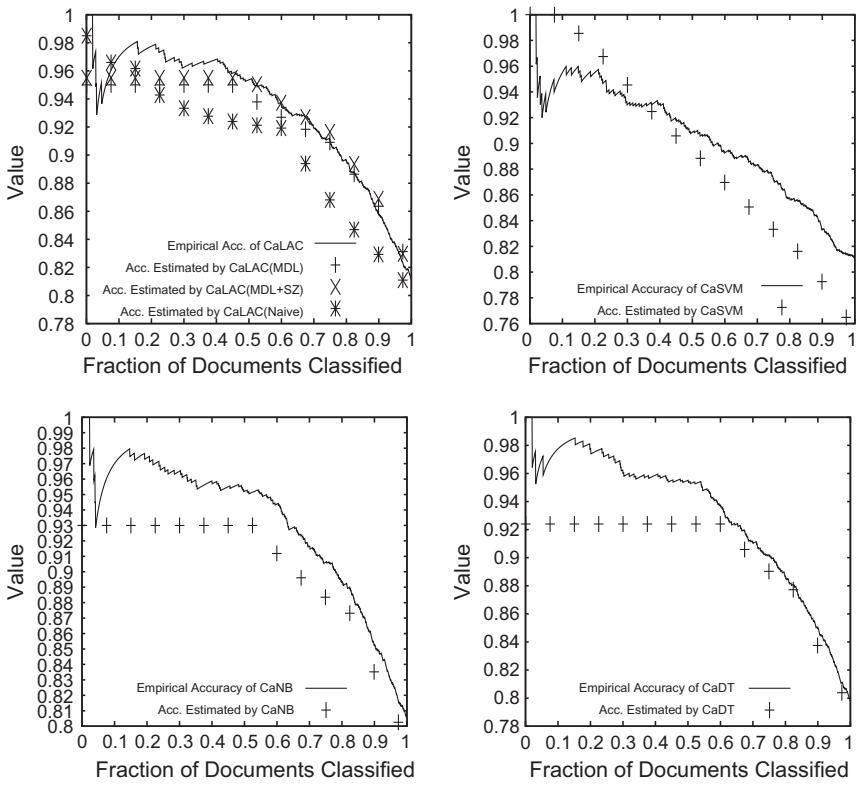

Fig. 5. Accuracy estimated by calibrated classifiers.

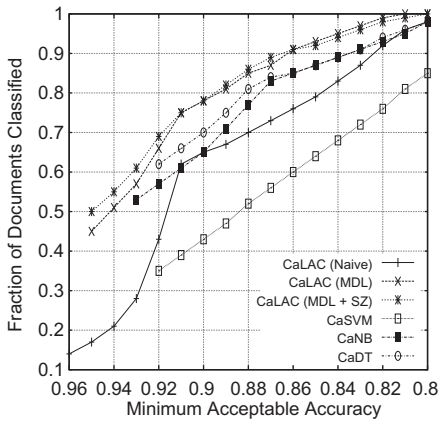

Fig. 6. Comparing calibrated classifiers.

The total cost invested in generating a request (including the mail cost), is \$0.68 per piece mailed. Thus, PVA wants to maximize net revenue by soliciting only individuals that are likely to respond with a donation. The KDD-98 dataset contains information about individuals that have (or have not) made charitable donations in the past. The provided training data consists of 95,412 examples, and the provided test set consists of 96,367 instances. Each example/instance corresponds to an individual, and is composed of 479 features. The training data has an additional field that indicates the amount of the donation (a value\$0 indicates that the individual has not made a donation). From the 96,367 individuals in the test set, only 4872 are donors. If all individuals in the test set were solicited, the total profit would be only \$10,547. On the other hand, if only those individuals that are donors were solicited, the total profit would be \$72,764. Thus, the classifier must choose which individuals to solicit a new donation from. According to [29], the optimal net maximization strategy is to solicit an individual

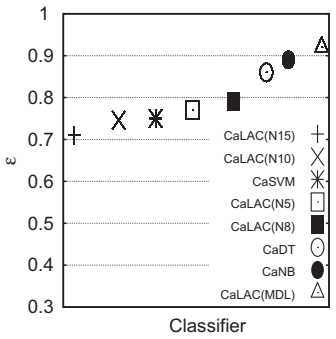

Fig. 7. Comparing calibrated classifiers in terms of ϵ.

x if and only if $\hat{p}(\text{donate}|x) > \frac{0.68}{y(x)}$, where $y(x)$ is the expected amount donated by x.[5] Thus, in addition to calculating $\hat{p}(\text{donate}|x)$, the classifier must also estimate $y(x)$.

5.2.1. Estimating the donation amount and $\hat{p}(\text{donate}|x)$

For this application, each donation amount (i.e.,\$200, \$199,...,\$1, and \$0) is considered as a class. Thus, for an individual x, rules of the form $\mathcal{X} \rightarrow y$ (with $\mathcal{X} \subseteq x$) are extracted from $\mathcal{D}$, and Eq. (2) is used to estimate the likelihood of each amount (i.e., $\hat{p}(y = \$200|x),\ \hat{p}(y = \$199|x), \ldots, \hat{p}(y = \$|x)$). The donation amount, $y(x)$, is finally estimated by a linear combination of the probabilities associated with each amount, as shown in Eq. (5). The probability of donation, $\hat{p}(\text{donate}|x)$, is simply given by $1 - \hat{p}(y = \$|x)$. All classifiers to be included in our comparison use this procedure in order to estimate the donation amount.

$$y(x) = \sum_{i=\$0}^{\$200} i \times \hat{p}(y = i|x) \tag{11}$$

5.2.2. Calibrating the classifiers

For the Naive Binning mechanism, we evaluated four different configurations, with 5, 8, 10, and 15 fixed-sized bins, which are, respectively, referred to as CaLAC (N5), CaLAC (N8), CaLAC (N10) and CaLAC (N15). We compared the calibration achieved by each configuration against the calibration using the MDL-based entropy minimization mechanism, CaLAC (MDL), in terms of ϵ. Other calibrated classifiers, CaSVM, CaNB and CaDT are also included in our comparison. The calibration degrees achieved by specific calibration mechanisms are shown in Fig. 7. CaSVM, CaLAC (N15) and CaLAC (N10) achieved the lowest calibration degrees. This is because the corresponding calibration mechanisms overfit the training data (i.e., for Naive Binning too many bins incur small-sized bins for which the corresponding accuracy may not be reliable). CaLAC (MDL) and CaNB are the best performers, achieving ϵ values as high as 0.94. For now on, we will use CaLAC (N8) as the representative of the classifier calibrated by the Naive Binning mechanism. We also performed a comparison involving different calibration mechanisms. Table 5 shows the degree of calibration achieved by these calibration mechanisms when applied to different classifiers. As can be seen, in terms of calibrating SVMs, Platt Scalling and Isotonic Regression performed much better than our proposed MDL-based entropy minimization mechanism. For Naive Bayes and Decision Trees, Isotonic Regression and our MDL-based entropy minimization calibration mechanism were the best performers in terms of ϵ. Finally, for LAC, our MDL-based calibration mechanism achieved the highest degree of calibration. Next we will analyze the effectiveness of classifiers with different calibration degrees for net revenue optimization.

5.2.3. Results

We used profit as the primary metric for assessing the effectiveness of the classifiers for net revenue optimization. For assessing the accuracy of probability estimates, we use the mean squared error (MSE), which is defined in Eq. (12), where $T(c_i|x)$ is 1 if the class of x is c_i, and 0 otherwise.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (T(c_i|x) - \hat{p}(c_i|x))^2 \tag{12}$$

Table 6 shows the effectiveness of each classifier. In all cases, the differences in profit are much more accentuated than the differences in MSE. As it can be seen, LAC achieved the lowest profit (which is slightly superior than soliciting all individuals), and this is because it was not calibrated yet. For the same reason, LAC was also the worst performer in terms of MSE. Calibrated classifiers CaLAC (N8) and CaSVM showed similar performance in terms of profit. According to [3], the poor per-

[5] The basic idea is to solicit a person x for whom the expected return $\hat{p}(\text{donate} - x)y(x)$ is greater than the cost of mailing the solicitation.

Table 5
Comparing different calibration mechanisms.

Classifier	Calibration	ϵ
SVM	Platt Scaling	0.74
	Isotonic Regression	0.73
	MDL-based	0.69
DT	Curtailment	0.86
	MDL-based	0.88
	Isotonic Regression	0.88
NB	*m*-Estimation	0.88
	Isotonic Regression	0.88
	MDL-based	0.92
LAC	Binning ($n = 8$)	0.79
	Isotonic Regression	0.86
	MDL-based	0.92

Table 6
Comparing classifiers in terms of profit and MSE.

Classifier	Profit	MSE
LAC	\$11,097	0.0961
CaLAC (N8)	\$12,442	0.0958
CaLAC (MDL)	\$14,902	0.0934
CaLAC (MDL + SZ)	\$**14,990**	0.0935
CaSVM	\$12,969	0.0958
CaNB	\$14,682	0.0952
CaDT	\$14,190	0.0953

formance of CaSVM is due to overfitting. CaDT and CaNB are again in close rivalry. Calibrating LAC using the entropy minimization mechanism is very profitable, and the corresponding classifiers, CaLAC (MDL) and CaLAC (MDL + SZ), are the best performers.

5.3. Learning to rank – the OHSUMED collection

For this application we used a dataset called OHSUMED, which is one of the collections composing the LETOR 3.0 Benchmark [17]. OHSUMED is a subset of MEDLINE that is a database on medical publications. OHSUMED has 106 queries. For each query is associated a number of documents and their respective relevance with respect to the query (i.e., definitely, possibly, or not relevant). There are a total of 16,140 query-document pairs with relevance judgments, and 45 extracted features. The classifier must approximate the relevance of each document. We performed 5-fold cross-validation and the final result represents the average of the five runs (all results to be presented were found statistically significant at the 95% confidence level when tested with a two-tailed paired *t*-test).

5.3.1. Estimating the relevance of documents

For this application, each level of relevance is considered a class (i.e., 0 for irrelevant, 1 for possibly relevant, and 2 for definitely relevant). Thus, for a document x, rules of the form $\mathcal{X} \rightarrow$ relevance (with $\mathcal{X} \subseteq x$) are extracted from $\mathcal{D}$, and Eq. (2) is used to estimate the likelihood of each relevance (i.e., $\hat{p}(\text{relevance=0}|x)$, $\hat{p}(\text{relevance = 1}|x)$, and $\hat{p}(\text{relevance} = 2|x)$). The relevance for document x, denoted as *rank*, is finally estimated by a linear combination of the probabilities associated with each relevance level, as shown in Eq. (13).

$$\text{rank} = \sum_{i=0}^{2} (i \times \hat{p}(\text{relevance} = i|x)) \tag{13}$$

5.3.2. Calibrating the classifier

We calibrated LAC using the MDL-based entropy minimization mechanism. Next we analyse the benefits of CaLAC (MDL), and how it improves ranking performance.

5.3.3. Results

Our goal is to evaluated how calibration improves ranking performance of LAC, and thus we performed a direct comparison with CaLAC (MDL). The comparison also includes state-of-the-art learning to rank methods. We used MAP (mean average precision)[6] for assessing ranking performance. Table 7 shows MAP numbers obtained by various learning to rank methods

[6] Average precision is the average of the precision scores at the rank locations of each relevant document. Mean average precision is the mean of the average precision scores for a group of queries.

Table 7
MAP numbers for the OHSUMED collection. Best results, including statistical ties, are shown in bold.

Trial	LAC	CaLAC	R-SVM	R-Boost	ListNet	AdaRank	SVMMAP
1	0.352	**0.361**	0.304	0.332	0.346	0.344	0.342
2	**0.463**	**0.469**	0.447	0.445	0.450	0.446	0.454
3	0.460	0.464	**0.465**	0.456	0.461	**0.469**	0.462
4	0.521	**0.529**	0.499	0.508	0.511	0.514	0.518
5	**0.482**	**0.485**	0.453	0.464	0.461	0.471	0.450
Average	0.456	**0.462**	0.433	0.441	0.446	0.449	0.445

(R-SVM [28], R-Boost [9], ListNet [1], AdaRank [27], SVMMAP [13]) using the OHSUMED collection. CaLAC is the best performer in four, out of five trials (it is significantly superior than LAC in 2 out of 5 trials). Further, it is also the best overall performer. This result suggests that calibrated class membership probabilities are a way to improve the ranking performance of LAC.

6. Conclusions

Calibration is the property of estimating reliable membership probabilities. This property is important in many classification problems. Most of the existing classifiers, although accurate, are not well calibrated, due to the bias imposed by the accuracy maximization strategy that is typically adopted. We proposed a lazy associative classifier (LAC) which does not follow the accuracy maximization paradigm (i.e., it simply describes the training data, and then it uses this description to infer membership probabilities that are used for prediction). We show that LAC is usually better calibrated than other classifiers. To further calibrate LAC, some mechanisms become necessary. We have proposed calibration mechanisms based on learning a mapping from original estimates to calibrated estimates. These mechanisms discretize the probability space into a set of bins, and for each bin it is associated a calibrated probability. One of the proposed mechanisms greatly differs from other existing approaches, because instead of using bins with pre-specified boundaries, it automatically finds the boundaries that minimize the entropy in each bin. We evaluate the proposed mechanisms by comparing them with other calibration mechanisms. The evaluation was carried using real-world applications. We showed that the entropy minimization mechanism, specially when coupled with LAC, provides the best overall results. As future work, we intend to move forward by investigating other calibration mechanisms and other application scenarios where calibration is important. Currently, we are studying the use of our calibration mechanisms in information retrieval applications that rely on ranking.

Acknowledgements

This research was supported by UOL (http://www.uol.com.br) through its UOL Bolsa Pesquisa Program (Grant No. 20080131200100), CNPq, Capes, Fapemig (Project No. 14281), INCTWeb (Grant No. 573871/2008-6), and InfoWeb (Grant No. 550874/2007-0).

References

[1] Z. Cao, T. Qin, T. Liu, M. Tsai, H. Li, Learning to rank: from pairwise approach to listwise approach, in: International Conference on Machine Learning, ACM, 2007, pp. 129–136.
[2] B. Cestnik, Estimating probabilities: a crucial task in machine learning, in: European Conference on Artificial Intelligence, 1990, pp. 147–149.
[3] I. Cohen, M. Goldszmidt, Properties and benefits of calibrated classifiers, in: European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag Inc., 2004, pp. 125–136.
[4] G. Cong, A.H. Tung, X. Xu, F. Pan, J. Yang, FARMER: finding interesting rule groups in microarray data, in: International Conference on Management of Data, ACM, 2004, pp. 143–154.
[5] J. Cussens, Bayes and pseudo-Bayes estimates of conditional probabilities and their reliability, in: European Conference on Machine Learning, Springer-Verlag Inc., 1993, pp. 136–152.
[6] A. Dawid, The well calibrated Bayesian, Journal of the American Statistical Association 77 (379) (1982) 605–613.
[7] M. DeGroot, S. Fienberg, The comparison and evaluation of forecasters, Statistician 32 (1982) 12–22.
[8] U. Fayyad, K. Irani, Multi interval discretization of continuous-valued attributes for classification learning, in: International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[9] Y. Freund, R. Iyer, R. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2003) 933–969.
[10] J. Friedman, T. Hastie, R. Tibishirani, Additive logistic regression: a statistical view of boosting, Annals of Statistics 2 (38) (2000).
[11] M.S. Gibbs, G.C. Dandy, H.R. Maier, A genetic algorithm calibration method based on convergence due to genetic drift, Information Sciences 178 (14) (2008) 2857–2869.
[12] Z. Jiang, Y.-X. Huang, Parametric calibration of speed–density relationships in mesoscopic traffic simulator with data mining, Information Sciences 179 (12) (2009) 2002–2013.
[13] T. Joachims, Optimizing search engines using clickthrough data, in: Intrnational Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 133–142.
[14] T. Joachims, Training linear SVMs in linear time, in: International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 217–226.
[15] J. Kruskal, Nonmetric multidimensional scaling: a numerical method, Psychometrika 29 (2) (1964) 115–129.
[16] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, in: International Conference on Knowledge Discovery and Data Mining, ACM, 1998, pp. 80–86.
[17] Y. Liu, J. Xu, T. Qin, W. Xiong, H. Li, LETOR: benchmark dataset for research on learning to rank for information retrieval, in: L2R SIGIR Workshop, 2007.

[18] A. Niculescu-Mizil, R. Caruana, Obtaining calibrated probabilities from boosting, in: Conference on Uncertainty in Artificial Intelligence, AUAI, 2005, pp. 413–420.
[19] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in: International Conference on Machine Learning, 2005, pp. 625–632.
[20] J. Platt, Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, Advances in Large Margin Classifiers (1999) 61–74.
[21] J. Quinlan, C4.5: Programs for Machine Learning, M. Kaufmann, 1993.
[22] J. Rissanen, Stochastic complexity and modeling, Annals of Statistics 14 (1986) 1080–1100.
[23] A. Veloso, H. Almeida, M. Gonçalves, W. Meira, Learning to rank at query-time using association rules, in: International Conference on Research and Development in Information Retrieval, ACM, 2008, pp. 267–274.
[24] A. Veloso, W. Meira Jr., M. Zaki, Calibrated lazy associative classification, in: Brazilian Symposium on Databases, SBC, 2008.
[25] A. Veloso, W. Meira Jr., M.J. Zaki, Lazy associative classification, in: International Conference on Data Mining, IEEE, 2006, pp. 645–654.
[26] A. Veloso, W. Meira, M. Cristo, M. Gonçalves, M. Zaki, Multi-evidence, multi-criteria, lazy associative document classification, in: Conference on Information and Knowledge Management, ACM, 2006, pp. 218–227.
[27] J. Xu, H. Li, Adarank: a boosting algorithm for IR, in: International Conference on Research and Development in Information Retrieval, ACM, 2007, pp. 391–398.
[28] Y. Yue, T. Finley, F. Radlinski, T. Joachims, A support vector method for optimizing average precision, in: International Conference on Research and Development in Information Retrieval, ACM, 2007, pp. 271–278.
[29] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers, in: International Conference on Machine Learning, 2001, pp. 609–616.
[30] B. Zadrozny, C, Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 694–699.