Mach Learn
DOI 10.1007/s10994-010-5170-2

Bayesian instance selection for the nearest neighbor rule

Sylvain Ferrandiz · Marc Boullé

Received: 5 December 2005 / Revised: 30 November 2009 / Accepted: 26 January 2010
© The Author(s) 2010

Abstract The nearest neighbors rules are commonly used in pattern recognition and statistics. The performance of these methods relies on three crucial choices: a distance metric, a set of prototypes and a classification scheme. In this paper, we focus on the second, challenging issue: instance selection. We apply a maximum a posteriori criterion to the evaluation of sets of instances and we propose a new optimization algorithm. This gives birth to Eva, a new instance selection method. We benchmark this method on real datasets and perform a multi-criteria analysis: we evaluate the compression rate, the predictive accuracy, the reliability and the computational time. We also carry out experiments on synthetic datasets in order to discriminate the respective contributions of the criterion and the algorithm, and to illustrate the advantages of Eva over the state-of-the-art algorithms. The study shows that Eva outputs smaller and more reliable sets of instances, in a competitive time, while preserving the predictive accuracy of the related classifier.

Keywords Nearest neighbor · Instance selection · Voronoi tesselation · Maximum a posteriori

1 Classification by the nearest neighbor

Supervised algorithms that learn classifiers input a finite sample $\{x_n, y_n\}$ of N instances $\{x_n\}$ and their respective labels $\{y_n\}$. The nearest neighbor rule (Fix and Hodges 1951; Cover and Hart 1967) classifies any previously unseen instance according to a vote among its nearest neighbor(s) in a set of prototypes derived from the sample. Building such a rule relies on the choice of a set of prototypes, the definition of a distance metric and the choice of a classification scheme.

Editor: David W. Aha.

S. Ferrandiz (✉) · M. Boullé
Orange Labs, 2, avenue Pierre Marzin, 22300 Lannion, France
e-mail: sylvain.ferrandiz@orange-ftgroup.com

M. Boullé
e-mail: marc.boulle@orange-ftgroup.com

Published online: 29 May 2010

1.1 Construction and selection of prototypes

The question of the construction of a set of prototypes can be tackled in many different ways. Following (Liu and Motoda 2001), we distinguish two different approaches, depending on whether the prototypes are derived from the instances or simply selected from the instances.

In (Salzberg 1991), instances are numerical tuples and prototypes are defined as "generalized exemplars". Such an exemplar is an axis-parallel hyperrectangle. An extension to the case of tuples composed of both numerical and categorical values is the aim of (Wettschereck and Dietterich 1995). Quantization methods define prototypes that are linear combinations of instances, considering the case of instances embedded in $\mathbb{R}^d$. For example, in (Chang 1991), an algorithm in which each instance is initially considered as a prototype is introduced: the pair of nearest instances of the same class are linearly combined into a single prototype, this process being iterated until classification accuracy starts to decrease.

Generalized exemplars are examples of prototypes whose representation differs from the initial representation of instances. Quantization exploits the algebraic structure of $\mathbb{R}^d$ and thus outputs prototypes within $\mathbb{R}^d$ but outside the set of instances. Generalized exemplars and Quantization are examples of prototype *construction* methods. We do not consider this approach in this article and focus instead on instance *selection*: the output set of prototypes is included in the original set of instances.

In practice, another distinction matters: whether the number of prototypes is an input or an output of the algorithm. In the first case, the number of prototypes is provided by the user, whereas in the second case, the algorithm selects the best one. For example, Linear Vector Quantization (LVQ) improves the position of K initial prototypes in $\mathbb{R}^d$ (Kohonen 2001). The initial prototypes may result from applying the K-means algorithm (MacQueen 1967) in each class separately or sampling from each class. The number K of prototypes is an input to LVQ and is set according to prior knowledge or using a validation set.

In this article, we consider the topic of instance selection, where:

– the set of prototypes is a subset of the set of instances (and we will use the terms "prototype" and "instance" interchangeably),
– the number of prototypes is an output of the algorithm.

1.2 Measuring similarities

In this article, we consider instance selection algorithms that take as input any distance metric, disregarding the representation of the instances (*i.e.* whether they are numerical, categorical, tuples, strings, graphs, etc). The selection process is guided by the pairwise computed similarities, not by the instances' features at all.

There is a great deal of work for designing different distance metrics for different kinds of features. When instances $\{x_n\}$ live in a real finite-dimensional space $\mathbb{R}^d$, the Euclidean and related L_p ($p \geq 1$) metrics are widely used. If needed, the Mahalanobis distance (Duda et al. 2001) takes into account the correlation between any two features. The Dynamic Time Warping heuristic applies to instances that are numerical sequences of possibly different sizes (Berndt and Clifford 1996). For finite and discrete multi-dimensional sets, which are the product of d finite alphabets, the Hamming metric is a popular metric. Since this metric becomes less reliable as the number of symbols increases, similarity measures based on frequencies have been proposed (Stanfill and Waltz 1986). These metrics are guided by the supervised classification task as their definition takes into account the labels. In order to deal with representations composed of both numerical and categorical measurements, weighting schemes can be applied in order to deal with scaling effects (Wilson and Martinez 1997a). When instances are strings or graphs, edit distances rely on editing operations in order to measure the similarity of two strings or graphs. For example, the Levenshtein distance makes use of substitution and insertion to compare two strings (Levenshtein 1966). When objects are represented by graphs, the problem is that of graph matching (Bunke 2000).

Every algorithm presented in this article applies to any distance metric.

1.3 Classification scheme

Once a similarity measure and a set of instances are selected, the nearest neighbors rules rely on the following classification scheme. For any instance x to be classified, a neighborhood of x is defined and the labeling is put to a vote among the neighbors of x. Such a scheme thus relies on the definition of a notion of neighborhood and a voting procedure.

In the K-NN rule (Fix and Hodges 1951), the neighborhood of x is defined as the set of the K nearest instances of x. In the Parzen window approach (Parzen 1962), the neighborhood is defined as the set of instances whose distance to x is under a given threshold h, called the *width* of the window. The winner-takes-all voting procedure is often adopted for the K-NN rule and when using Parzen windows. It assigns to x the most frequent label in the neighborhood of x. If closer instances are weighted more heavily, the decision is smoother. Kernel rules adopt such a classification scheme (Devroye et al. 1996). They extend the Parzen window approach and the width acts as a smoothing factor.

Instance selection algorithms usually apply a K-NN rule with a winner-takes-all voting procedure. This will be referred to as the *K-NN classification scheme*.

1.4 Proposed approach for instance selection

In (Devroye et al. 1996), the authors introduce a tool they call *relabeling*. It works as follows. Assume that we have a classifier g_D where D is the data $\{x_n, y_n\}$. For example, this classifier results from the application of the K-NN rule. Let us define the new labels $z_n = g_D(x_n)$. The relabeling method for classification applies the 1-NN rule to the new data $\{x_n, z_n\}$. In the chosen example, it means that a new instance x is classified according to a winner-takes-all procedure in the neighborhood of its nearest instance, and not in its own neighborhood.

We introduce a new classification scheme for instance selection named the *Voronoi-Based Relabeling scheme*, or VBR scheme for short, which is a relabeling method relying on Voronoi partitions. This is a two-stage process closely related to instance selection. Indeed, if we perform no selection, the VBR scheme reduces to the 1-NN rule. In this article, we exploit some properties of this new scheme and we propose a new method to optimize instance selection for this scheme.

Before defining the VBR scheme, let us fix some notations. We denote $\mathcal{U} = [\![1, N]\!]$ the set of the N *statistical units*. As an example, for the analysis of detailed phone call records, the household stands for the statistical unit. Each statistical unit n is subject to a set of measurements and the resulting tuple is denoted x_n. This is the *instance* of n and we denote $X : n \mapsto x_n$ the *descriptive feature*. We denote $\mathcal{L} = [\![1, J]\!]$ the set of the J possible *labels*. A label $y_n \in \mathcal{L}$ is associated with each unit $n \in \mathcal{U}$ and we denote by $Y : n \mapsto y_n$ the resulting *target feature*.

A *kernel* is an application δ that assigns a positive real value to every pair of units (*cf.* Scholkopf and Smola 2001). It stands for the given distance metric. While computing similarities certainly depend on the representation of the instances in practice, we make the notation independent of the instances in order to illustrate that all the algorithms presented here depend on the kernel only and not on the representation of the statistical units. That is why we will use n and x_n and the terms "unit" and "instance" interchangeably in the next sections.

Given a kernel δ, any finite set of units defines a Voronoi tesselation. Let H be a finite subset of $\mathbb{N}$. For $k \in H$, the *Voronoi cell* $V_\delta(k, H)$ is the set

$$V_\delta(k, H) = \left\{ n \in \mathbb{N};\ k = \arg\min_{k' \in H} \delta(n, k') \right\}. \tag{1}$$

The element k is called the *representative* or the *prototype* of its cell $V_\delta(k, H)$. The family of the Voronoi cells $V_\delta(H) = (V_\delta(k, H))_{k \in H}$ is the *Voronoi tesselation* associated with H.

From now on and implicitly, we assume that Voronoi tesselations are partitions. In practice, one has to deal with distance ties. We adopt the heuristic explained in (Devroye et al. 1996) and add a random component to the descriptive feature to circumvent the problem.

We can now define the VBR scheme more precisely: a new instance x is classified according to a winner-takes-all procedure in the cell of its nearest prototype. This scheme has several advantages:

- it is nonparametric
- it is better suited to low selection rates than the K-NN scheme (*cf.* Appendix A)
- instance selection for the VBR scheme is easily turned into a probabilistic hypothesis selection problem.

Indeed, let us define a *model* as a couple (H, Ψ) where H is a set of K instances and Ψ a matrix giving at the position (j, k) $(1 \le j \le J, 1 \le k \le K)$ the frequency of the label j in the cell k of $V_\delta(H)$. We denote $\mathcal{M}_\delta(\mathcal{U})$ the set of models. The problem of instance selection for the VBR scheme is now a problem of hypothesis selection: which model in $\mathcal{M}_\delta(\mathcal{U})$ is the best, with models defined as conditional probability distributions (*cf.* Fig. 1)?

It is noteworthy that the VBR scheme reduces to the 1-NN scheme if no instance selection is performed at all. The VBR scheme and instance selection are thus intrinsically linked.

When designing an instance selection algorithm, one needs a criterion for evaluating the models in $\mathcal{M}_\delta(\mathcal{U})$ and an efficient algorithm for browsing through this set. In order to fully exploit the properties of the relabeling tool, the criterion must:

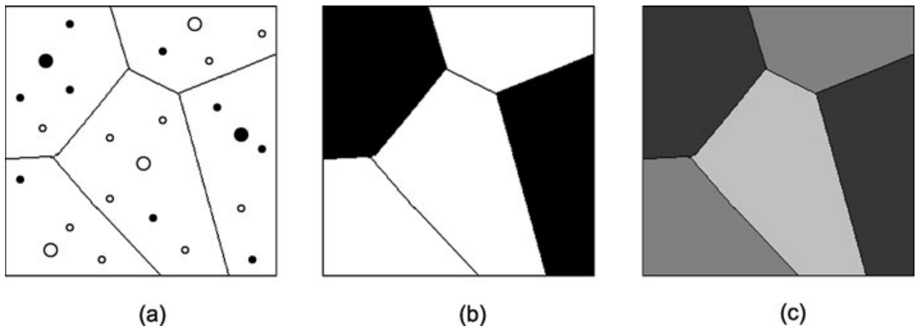


Fig. 1 Instance selection for the relabeling tool is a model selection problem with models defined as conditional probability distributions. Owing to the relabeling tool, the classification relies on more detailed information. (**a**) A Voronoi partition for a binary classification problem. (**b**) Straightforward classification consists in coloring the cell according to the label of the prototype. (**c**) Relabeling allows to take into account the frequency of every label in each cell when coloring the cell

– be nonparametric
– take into account the size of the set of prototypes
– consider the whole distribution of the labels in each cell.

In (Ferrandiz and Boullé 2006), we introduced a maximum a posteriori (MAP) criterion for evaluating such models. In this article, we propose a new optimization heuristic for finding the best set of prototypes according to this criterion. We benchmark the resulting instance selection method, named Eva.

1.5 Organization of the paper

The remainder of the article is organized as follows. We present an overview of the existing instance selection methods in Sect. 2 and cluster them into two groups: the global ones, which evaluate a set of prototypes as a whole, and the local ones, which rely on an individual measure of interestingness for each instance.

We introduce in Sect. 3 the criterion presented in (Ferrandiz and Boullé 2006), for the convenience of the reader. In Sect. 4, we describe a new heuristic with some optimization that exploits the properties of the criterion in order to increase its scalability. Eva, the resulting instance selection method, falls into the category of global evaluation methods.

In Sect. 5, we benchmark Eva on several real datasets. We conduct further experiments in order to evaluate the benefit of the optimization heuristic versus the criterion. The contribution of the heuristic is demonstrated in Sect. 6. The experiments in Sect. 7 analyze the behavior of instance selection methods and illustrate the contribution of Eva.

2 Instance selection

Given a set $\mathcal{U}$ of N instances with their respective labels and a kernel δ, an instance selection algorithm attempts to select the "best" instances for the nearest neighbor classifiers. In this section, we overview the field of instance selection heuristics.

We distinguish methods based on a local evaluation from methods based on a global evaluation. A *global* method defines an evaluation criterion for any subset of instances. In contrast, a *local* method relies on the definition of a criterion measuring the predictive interestingness of a single instance. The distinction is based on the evaluation criteria rather than the algorithms.

2.1 Global methods

In supervised classification, empirical risk minimization is a common evaluation paradigm. It makes use of the empirical risk as the evaluation criterion: for a given classifier f, each label y_n is compared with the label computed by f; the number of misclassifications over the number N of instances defines the *empirical risk* or *error rate*. The empirical risk associated with a subset H of $\mathcal{U}$ thus relies on the comparison between the label of each instance with the label of its nearest neighbor in H. Many instance selection methods attempt to minimize this risk.

The CNN (for Condensed Nearest Neighbor) selection method (Hart 1968) is one of them. CNN is incremental and makes several passes through the dataset. This method is *consistent*: the empirical risk of the resulting set of prototypes H is null. In order to obtain a minimal subset according to the consistency property, and under the hypothesis that H contains such a subset, the RNN rule (for Reduced Nearest Neighbor) proposed in (Gates 1972) removes from H every prototype which causes no new misclassification. The computational complexity is of $O(N^3)$, close to $O(N^2)$ in the average case.

The Delaunay triangulation provides another solution to the selection of a consistent subset (Toussaint and Poulsen 1975). When instances lie in a Euclidean space and the selected kernel is the Euclidean metric, for a given set of prototypes, two Voronoi cells define an edge if they share a common boundary. The resulting graph defines a Delaunay triangulation of the Euclidean space. Considering the Voronoi cells associated with $\mathcal{U}$, each cell of the partition contains a single instance, its prototype, which defines a vertex of the Delaunay graph. A vertex whose neighbors in the Delaunay graph have the same label can be discarded without any modification of the overall decision boundary. The resulting instance selection method is thus consistent.

Computing the Delaunay graph is intractable as the complexity of the algorithm grows exponentially with the dimension of the Euclidean space. Approximation schemes have been proposed (Toussaint et al. 1985). Indeed, the Delaunay graph can be substituted by any proximity graph, like the Gabriel graph or the relative neighborhood graph (Jaromczyk and Toussaint 1992). The computation of a Gabriel (or relative) neighborhood is still time-consuming ($O(N^2)$ operations needed to find the set of Gabriel or relative neighbors of an instance). Approximated neighborhoods, which can be computed more quickly, are introduced in (Bhattacharya et al. 2005). However, the consistency property is lost.

In place of the empirical risk, the following criterion is proposed in (Cameron-Jones 1995) to evaluate sets of prototypes:

$$e(H) = F(K, N) + K \log_2(J) + F(E, N - K) + E \log_2(J - 1), \tag{2}$$

where K is the number of prototypes in H and E is the number of instances misclassified by their nearest prototype in H. This criterion is rooted in the Minimum Description Length principle (Grünwald et al. 2005). The terms $F(K, N) + K \log_2(J)$ evaluate the description length of the model (*i.e.* the selection of K instances and their respective labels). The terms $F(E, N - K) + E \log_2(J - 1)$ evaluate the description length of the exceptions (*i.e.* the E misclassified instances and their respective labels).

In (Cameron-Jones 1995), the author proposes the following optimization heuristic. A first incremental step randomly browses through the set of instances and selects an instance if the value of the criterion decreases. Every instance is considered once. A second decremental step removes a prototype if the criterion is optimized. Finally, 1000 mutations are performed and evaluated. A mutation is triggered if the criterion decreases. A mutation consists in adding an instance to the set of prototypes, or removing a prototype or swapping an instance and a prototype. This heuristic will be denoted Explore(M), where M is the number of mutations. It has a complexity of $O(K_{max} N^2 + M N^2)$, with K_{max} the maximum size of a set of prototypes considered during the optimization.

2.2 Local methods

A Local method relies on an individual predictive interestingness measure instead of a global criterion. For example, the ENN rule (for Edited Nearest Neighbor) described in (Wilson 1972) considers the L nearest neighbors in $\mathcal{U}$ of each instance (typically $L = 3$). An instance is removed if it is misclassified by a majority vote among its neighbors. The complexity of the algorithm ENN(L) is of $O(LN^2)$. In practice, ENN can be viewed as a noise filtering rule. This method can be applied iteratively.

More recently, D. Aha and his colleagues proposed a series of algorithms: IB1 to IB5 (Aha et al. 1991; Aha 1992). The last two versions are out of the scope of this paper since they deal with feature selection. IB1 is an incremental nearest neighbor rule and serves as a baseline. IB2 is a limited version of the CNN rule, making a single pass through the dataset. IB3 enhances the CNN rule by introducing a notion of acceptability of a prototype.

The acceptability of a prototype is determined by comparing the bounds of two confidence intervals: if the lower bound on the accuracy is higher than the upper bound on the frequency of its label, the prototype is *acceptable*; if the upper bound on the accuracy is lower than the lower bound on the frequency of its label, the prototype is *poor*. The confidence level is set to 90% for acceptance and to 75% for dropping. The bounds are computed according to the following formula:

$$\frac{c + \frac{z^2}{2n} \pm \sqrt{\frac{c(1-c)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}. \tag{3}$$

For the accuracy of a prototype, n is the number of times its classification record has been updated since its introduction in the set of prototypes, c is the number of times an instance has been well-classified by it and z the confidence level. For the frequency of a label, n is the number of previously processed instances, c is the proportion of instances so far that are of this class and z the confidence level. Finally, the algorithm has a complexity of $O(N^2)$.

Another tool for measuring the interestingness of an instance, based on a notion of association, is presented in (Wilson and Martinez 1997b). For fixed L (typically $L = 3$), an instance x_{n_2} is an *associate* of x_{n_1} if x_{n_1} is one of the L nearest neighbors of x_{n_2}. This notion is used in a series of selection methods, from DROP1 to DROP5, from which DROP3 has been shown to perform best in (Wilson and Martinez 2000). The core algorithm is decremental, dropping an instance x_n if at least as many of its associates are still correctly classified without it (DROP1). The instances are sorted according to their distance to the nearest instance lying in another class and checked decreasingly (DROP2). Furthermore, the ENN(L) rule is applied as a preprocessing step in order to detect and remove mislabeled instances (DROP3). The overall complexity of the method is of $O(LN^2)$.

In (Sebban et al. 2002), a statistical test is defined. An instance is tested in order to know whether it contributes to the classification of its associates. This criterion is parametric and relies on the estimation of the density of the associated statistic. An incremental optimization heuristic is derived from the AdaBoost algorithm. The quickest version of the resulting selection method, named PSBoost2, has a complexity of $O(LN^2)$.

When local methods rely on a neighborhood defined by the L nearest neighbors, the local criterion is parametric. In order to obtain nonparametric criteria, it is possible to substitute the L nearest neighbors by the adjacent vertices of the instance in a proximity graph. For example, the work described in (Sanchez et al. 1997) results from the use of this trick.

A nonparametric notion of association is proposed in (Brighton and Mellish 2002), slightly modifying the one used in DROP algorithms. At each step, the algorithm considers the set H of previously selected instances. Instead of considering the L nearest neighbors in H of an instance in H for a fixed L, they proposed to take into account the instances that are nearer than the nearest instance with another label. The neighborhood of an instance thus contains instances with the same label. The definition of an associate in H of an instance in H remains the same. At the end of the step, instances in H whose neighborhood is bigger than the set of their associates are removed. This step is iterated till no removal is triggered. The algorithm ENN(L) is run as an initial preprocessing step. While the core algorithm is nonparametric, and has a complexity of $O(N^3)$, close to $O(N^2)$ in the average case, the application of ENN makes the overall algorithm, denoted ICF(L), parametric.

2.3 Summary

Local methods rely on an individual notion of "interestingness". The experiments in (Brighton and Mellish 2002) show that ICF(3) performs best among the local methods. We performed similar experiments, the results of which, not reported here, confirm that ICF outperforms alternative local methods (ENN, CNN, IB3 and DROP3). It is thus the best candidate for representing local methods.

Global methods evaluate sets of prototypes as a whole. Methods based on empirical risk minimization fail: the solution overfits the data, as it is well-known in the field of statistical learning (Vapnik 1996). A solution to this problem consists in adding a regularization term to the fitness term. This is what the MDL criterion of Explore does, in a nonparametric manner. Explore is thus the best candidate for representing global methods.

3 Maximum a posteriori evaluation of a set of prototypes

In (Ferrandiz and Boullé 2006), we introduced a new criterion for evaluating sets of prototypes. In this section, we synthesize this work for the convenience of the reader. We first illustrate the intuition supporting the criterion. Second, we place the work into the Bayesian framework. Then, we introduce the criterion and discuss its properties.

3.1 The intuition

In Sect. 1, we proposed to consider the problem of instance selection for the VBR scheme as a problem of model selection: find the best couple (H, Ψ) where H is the set of selected instances and Ψ the matrix of the frequencies of the labels in the cells of $V_\delta(H)$. We aim to define a criterion $c : \mathcal{M}_\delta(\mathcal{U}) \to \mathbb{R}$ (*cf.* Sect. 1.4 for the notations) for evaluating which model is the "best". This criterion should:

– be nonparametric
– take into account the size of the set of prototypes
– consider the whole distribution of the labels in each cell.

The optimization of such a criterion leads to a nonparametric instance selection algorithm for the VBR scheme, with the number of prototypes being an output of the method. By taking into account every label in each cell and not only the label of the prototype, the evaluation leads to finer models (probability distributions and not classifiers). In (Ferrandiz and Boullé 2006), we introduced a criterion fulfilling this ambition. Adopting a Bayesian point of view, the criterion is composed of

1. a prior term capturing the complexity of a set of prototypes and depending on its size
2. a likelihood term capturing the heterogeneity of the distributions of the labels.

3.2 The Bayesian framework

A classical paradigm in statistics and machine learning consists in selecting the most probable model M given the data D. This is the maximum a posteriori (MAP) paradigm, as it relies on the definition of an a posteriori probability $P(M/D)$ on a set of candidate models. The approach consists in applying the Bayes' formula and writing $\arg\max_M p(M/D) = \arg\max_M p(M)p(D/M)$. A statistician then defines a prior $p(M)$, a likelihood $p(D/M)$ and the quality of the model is evaluated by the product $p(M)p(D/M)$.

In (Ferrandiz and Boullé 2006), we adapted the approach to the particular case of instance selection, where the set of models $\mathcal{M}_\delta(\mathcal{U})$ depends on the data. Data consists here in the set of instances $\mathcal{U}$, the kernel δ and the target feature Y. The models are couples (H, Ψ), where H is the set of selected instances and Ψ being the matrix of conditional probability distributions. Bayes' formula allows us to write

$$\begin{aligned} p(H, \Psi/D) &= p(H, \Psi/Y, \delta, \mathcal{U}) \\ &= \frac{p(H, \Psi, Y, \delta, \mathcal{U})}{p(Y, \delta, \mathcal{U})} \\ &= \frac{p(\delta, \mathcal{U}) p(M/\delta, \mathcal{U}) p(Y/M, \delta, \mathcal{U})}{p(Y, \delta, \mathcal{U})}. \end{aligned} \tag{4}$$

If we cancel the terms not depending on $M = (H, \Psi)$, the MAP decision can be written

$$\arg\max_M \ p(M/\delta, \mathcal{U}) p(Y/M, \delta, \mathcal{U}). \tag{5}$$

By conditioning on $\mathcal{U}$, the modeling of the instances is pulled aside and we focus on the relationship between the instances and the labels. We have to define a prior distribution $p(\cdot/\mathcal{U}, \delta)$ on the set of models and a likelihood $p(\cdot/M, \mathcal{U}, \delta)$ on the set of target features in order to obtain an effective MAP decision.

First, the probability of a model $M = (H, \Psi)$ with K prototypes is written as the product of the probability of K and the probability of M given K. More precisely:

$$p(M/\delta, \mathcal{U}) = p(K, M/\delta, \mathcal{U}) = p(K/\delta, \mathcal{U}) p(M/K, \delta, \mathcal{U}). \tag{6}$$

We apply the Bayes' formula to the last term and we write

$$p(M/K, \delta, \mathcal{U}) = p(H/\delta, \mathcal{U}) p(\Psi/H, \delta, \mathcal{U}). \tag{7}$$

Second, we adopt an independence hypothesis of the labels between cells. This hypothesis is weaker than the usual i.i.d. assumption on the dataset. We write:

$$p(\Psi/H, \delta, \mathcal{U}) = \prod_{k=1}^{K} p(\Psi_k / V_\delta(k, H), \delta, \mathcal{U}) \tag{8}$$

and

$$p(Y/M, \delta, \mathcal{U}) = \prod_{k=1}^{K} p(Y_k / \Psi_k, V_\delta(k, H), \delta, \mathcal{U}), \tag{9}$$

where Ψ_k is the conditional distribution of the labels in the cell $V_\delta(k, H)$ (*i.e.* the k^{th} row of Ψ) and Y_k denotes the restriction of Y to the cell $V_\delta(k, H)$.

3.3 The probabilistic criterion

According to the MAP decision process, a model $M = (H, \Psi)$ is evaluated by the a posteriori probability $p(M/Y, \delta, \mathcal{U})$. According to the previous section, the definition of the primary a posteriori probability relies on the definition of four probabilities. In (Ferrandiz and Boullé 2006), these four probabilities are made explicit, adopting uniform distributions.

First, the number K of selected instances is a number greater than 1 and lower than N and we set

$$p(K/\delta, \mathcal{U}) = \frac{1}{N}. \tag{10}$$

Second, we consider H as a K-combination with repetitions. The number of such combinations is $\binom{N+K-1}{K}$ and we set

$$p(H/K, \delta, \mathcal{U}) = \frac{1}{\binom{N+K-1}{K}}. \tag{11}$$

Third, in the k^{th} cell, owing to the dependency, we consider a restricted support for the possible conditional probabilities and take into account rational probabilities only. More formally the support is $\{(\frac{N_{k1}}{N_k}, \ldots, \frac{N_{kJ}}{N_k})\}$, where N_k is the number of instances in the k^{th} cell. The cardinality of the support is $\binom{N_k+J-1}{J-1}$ and, adopting a uniform prior, we set

$$p(\Psi_k / V_\delta(k, H), \delta, \mathcal{U}) = \frac{1}{\binom{N_k+J-1}{J-1}}. \tag{12}$$

Fourth, according to the dependency, it remains to specify the label of each instance in each cell given the partition $V_\delta(H)$ and the conditional probabilities $\Psi_k = (\frac{N_{k1}}{N_k}, \ldots, \frac{N_{kJ}}{N_k})$. For the k^{th} cell, the problem is the same as putting the instances of the cell in J boxes, under the condition that the j^{th} box contains N_{kj} elements ($1 \leq j \leq J$). The multinomial coefficient gives the exact number of possibilities and we obtain

$$p(Y_k / \Psi_k, V_\delta(k, H), \delta, \mathcal{U}) = \frac{1}{\frac{N_k!}{N_{k1}! \ldots N_{kJ}!}}. \tag{13}$$

Finally, if we denote $c(M) = -\log p(M/Y, \delta, \mathcal{U})$, we aim to minimize:

$$c(H) = \log N + \log \binom{N+K-1}{K} + \sum_{k=1}^{K} \log \binom{N_k + J - 1}{J-1} + \log \frac{N_k!}{N_{k1}! \ldots N_{kJ}!}, \tag{14}$$

where K is the size of H, N_k is the number of instances lying in the k^{th} cell $V_\delta(k, H)$ and N_{kj} is the number of instances in the k^{th} cell with the label j ($1 \leq k \leq K$, $1 \leq j \leq J$).

It is worth noting that this criterion is *additive*: we can write

$$c(H) = c_1(K) + \sum_{k=1}^{K} c_2^{(k)}(H, Y), \tag{15}$$

where $c_1(K) = \log N + \log \binom{N+K-1}{K}$ and, for $1 \leq k \leq K$, $c_2^{(k)}(H, Y) = \log \binom{N_k+J-1}{J-1} + \log \frac{N_k!}{N_{k1}! \cdots N_{kJ}!}$. This property, shared by the empirical risk and Explore's criterion, will prove useful when designing our optimization algorithm. Finally, we denote

$$c_2(H, Y) = \sum_{k=1}^{K} c_2^{(k)}(H, Y). \tag{16}$$

3.4 Summary

The likelihood term, according to Stirling's approximation $\log x! \approx x \log x - x + O(\log x)$, behaves asymptotically as N times the conditional entropy of the distribution of the y_n's given the cluster assignment function:

$$\frac{1}{N}\sum_{k=1}^{K}\log\frac{N_k!}{N_{k1}!\cdots N_{kJ}!} \approx -\sum_{k=1}^{K}\sum_{j=1}^{J}\frac{N_{kj}}{N}\log\frac{N_{kj}}{N_k}. \tag{17}$$

The criterion thus evaluates the conditional probabilities with a finite-data entropy-related term balanced with a structural weight, which quantifies the complexity of the model by taking into account the number K of selected instances. This criterion is nonparametric. This results from the adaptation of a MAP approach to the case of data-dependent models, without making the iid assumption on the sample, without defining a generative process of the instances, without defining a generative process of the labels given the instances and without adopting a parametric prior. This prior is hierarchical and depends on the data through the set of models (Robert 2001).

4 A new optimization heuristic

In this section, we describe a new optimization algorithm to find the best set of instances. First, it consists of a bottom-up greedy heuristic, the complexity of which can be reduced by exploiting the fact that the criterion is additive. Second, a heuristic which repeatedly applies the greedy algorithm is designed according to the Variable Neighborhood Search meta-heuristic. We name the resulting algorithm VNSGreedy. This algorithm applies to the optimization of the empirical risk and the criterion given by Eq. 2, as they are additive.

4.1 Greedy optimization of a set of prototypes

The greedy heuristic Greedy(H) takes as input a set H of K prototypes (*cf.* Algorithm 1). Every subset resulting from the removal of an element in H is evaluated. Among those subsets, the winner is the one minimizing the criterion. This process is iterated and applied to every successive winner, until a singleton has been evaluated. The best encountered subset is returned. Note that this well-known heuristic has been applied to the problem of feature selection, giving rise to the Backward Sequential Elimination algorithm (Guyon and Elisseeff 2003).

The greedy method considers $O(K^2)$ subsets and each evaluation requires the search of the nearest prototype for each instance. A straightforward implementation of Greedy(H) has a complexity of $O(NK^3)$.

We now propose some optimizations for additive criteria. The complexity is reduced to $O(NK\log K)$ by exploiting the additive property of the criterion, as described next.

At each step, every removal of a prototype affects the instances of its cell only. At each step, the N instances are thus considered once, with a computational cost that we make a constant (*cf.* Algorithm 2).

First, if for each instance the prototypes are sorted according to their distance to this instance, the next nearest prototype is available at a constant cost. An initialization step is added to the greedy algorithm which consists in building the N sorted lists of size K. The complexity is of $O(NK\log K)$ and storing these N lists requires a memory space of

Algorithm 1: Greedy algorithm

```
Data: $X$ // Descriptive feature //, $Y$ // Target feature //, $\delta$ // Kernel //
Input: $H$ // Set of prototypes
Output: a subset $S_{best} \subset H$ such that $c(S_{best})$ is minimal among evaluated subsets
1 $S \leftarrow H$;
2 $S_{best} \leftarrow H$;
3 while $\#S > 1$ do
4     $s_0 \leftarrow \arg\min_{s \in S} c(S \setminus \{s\})$;
5     $S \leftarrow S \setminus \{s_0\}$;
6     if $c(S) < c(S_{best})$ then
7         $S_{best} \leftarrow S$;
8 return $S_{best}$;
```

$O(NK)$. By storing the previously removed prototypes, which have to be considered as unavailable, the lists can be updated on the fly and when needed. As each update occurs at most once during the algorithm, there are at most NK updates. Thus, building and maintaining the lists has an overall complexity of $O(NK \log K)$.

Second, owing to the additive property of the criterion, its value can be updated with a constant cost. Only the term $c_2(H, Y)$ of the criterion depends on the partition of the instances in the cells. When a prototype k is removed, the first thing to do is to subtract its contribution $c_2^{(k)}(H, Y)$ to the value of the criterion. Then, any instance n lying in the corresponding cell is associated with its next nearest prototype k_{next}. The parameters $N_{k_{next}}$ and $N_{k_{next} j_0}$, with j_0 the index of the label of the considered instance, are updated by a simple increment. The term related to the prototype k_{next} is then updated by adding the quantity $SwappingCost(n, k, k_{next}, H, Y) := \log(N_{k_{next}} + J) - \log(N_{k_{next} j_0} + 1)$.

To summarize, at each of the K steps of the algorithm, each instance is considered once. Finding its next nearest prototype and updating the value of the criterion is performed with a constant cost, given N sorted lists of size K and owing to the property of the criterion. Computing and storing the lists has a complexity of $O(NK \log K)$ and requires a memory space of $O(NK)$. Once the lists have been computed, the optimized greedy heuristic has a complexity of $O(KN)$. The overall complexity of the algorithm described in Algorithm 2 is thus of $O(NK \log K)$.

4.2 Variable neighborhood search

The greedy heuristic performs many evaluations quickly. It is then natural to think about a repeated application of this algorithm. We want to limit the number of applications of the greedy heuristic and to improve the solution with each new greedy evaluation. In other words, we attempt to make the most of the given training time. This is done according to the Variable Neighborhood Search (VNS) meta-heuristic (Hansen and Mladenovic 2001), which consists in applying the primary heuristic (*i.e.* the greedy one) to a neighbor of the solution. If the new solution is not better, a bigger neighborhood is considered. Otherwise, the algorithm restarts with the new best solution and a minimal size neighborhood. The process relies on the definition of a notion of a *solution neighborhood.*

We propose the following definition for the neighborhoods of a set of prototypes. Intuitively speaking, we define a neighbor H of a set of prototypes H_0 as a bigger set (*i.e.*

Algorithm 2: Optimized greedy algorithm

```
Data: $X$ // Descriptive feature //, $Y$ // Target feature //, $\delta$ // Kernel //, $N$ // Number of
      instances
Input: $H$ // Set of prototypes
Result: a subset $S_{best} \subset H$ such that $c(S_{best})$ is minimal among evaluated subsets
1  for $n = 1$ to $N$ do
2      $L_n \leftarrow$ the list of the prototypes, sorted with respect to their similarity with the $n^{th}$
       instance;
3  Mark every prototype as available;
4  $S \leftarrow H$;
5  $S_{best} \leftarrow H$;
6  $BestCost \leftarrow c(H)$;
7  $BestLabelCost \leftarrow c_2(H, Y)$;
8  for $t = K - 1$ to $1$ do
9      $StructuralCost \leftarrow c_1(t)$;
10     $LocallyBestLabelCost \leftarrow +\infty$;
11     for $s \in S$ do
12         $LabelCost \leftarrow BestLabelCost$;
13         $LabelCost \leftarrow LabelCost - c_2^{(s)}(H, Y)$;
14         for $n \in V_\delta(s, S)$ do
15             $s' \leftarrow$ the next available prototype nearest to $n$, found in $L_n$;
16             $LabelCost \leftarrow LabelCost + SwappingCost(n, s, s', H, Y)$;
17         if $LabelCost < LocallyBestLabelCost$ then
18             $LocallyBestLabelCost \leftarrow LabelCost$;
19             $s_{best} \leftarrow s$;
20     $S \leftarrow S \setminus \{s_{best}\}$;
21     $BestLabelCost \leftarrow LocallyBestLabelCost$;
22     Mark $s$ as unavailable;
23     if $StructuralCost + BestLabelCost < BestCost$ then
24         $BestCost \leftarrow StructuralCost + BestLabelCost$;
25         $S_{best} \leftarrow S$;
26 return $S_{best}$;
```

$H_0 \subset H$), resulting from first removing prototypes from H_0 and then adding instances. The number of removed and added instances is controlled by a rate $t \in [0, 1]$. More technically, a *neighbor* of H_0 is every set $H = H_1 \sqcup H_2$ where $H_1 \subset H_0$ and H_2 is a set of instances lying in the cells of $V_\delta(H_0)$ associated with the prototypes in $H_0 \setminus H_1$. If $t \in [0, 1]$, the neighborhood $\mathcal{V}_t(H_0)$ is defined as the set of neighbors $H = H_1 \sqcup H_2$ of H_0 such that the rate of replaced elements in H_0 and the rate of selected instances in the associated cells is t (*cf.* Fig. 2).

If the solution is never improved by successively exploring neighborhoods of growing size, the algorithm stops when a neighbor of maximal size is evaluated (the size of a set of prototypes is bounded by the number N of instances). Instead of incrementally exploring the neighborhoods by adding only one prototype at a time, which is time consuming for poor improvement, we propose to use a series of selection rates $t_0, \ldots, t_{MaxDegree}$, where *MaxDegree* is determined by the user. This parameter quantifies the training time allowed

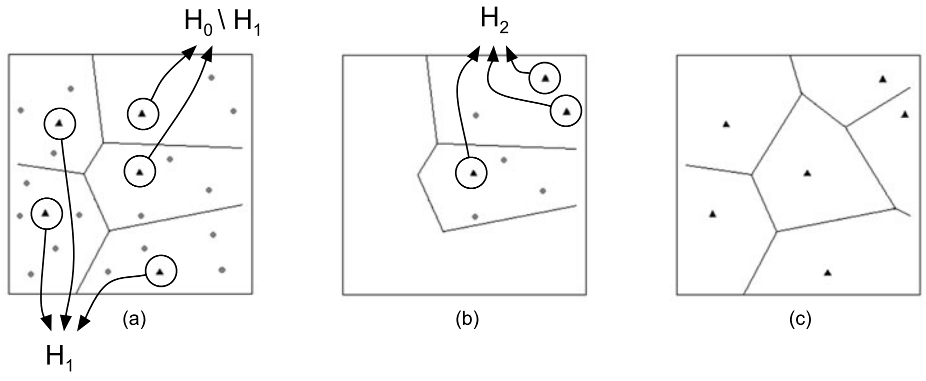


Fig. 2 Example of a neighbor of a set of prototypes for $t = 0.35$. (**a**) Partition of the instances in the cells. Two prototypes (i.e. 35% of the prototypes in H_0) are replaced. (**b**) 3 instances (i.e. 35% of the instances in the cells associated with the prototypes in $H_0 \setminus H_1$). (**c**) The partition associated with $H_1 \bigcup H_2$

Algorithm 3: VNSGreedy algorithm

```
Data: $X, Y, \delta$
Input: $MaxDegree$
Result: a set of prototypes, each prototype coupled with the most frequent label within
        its Voronoi cell
1  $S_{best} \leftarrow$ OptimizedGreedy($\mathcal{U}$);
2  $S \leftarrow S_{best}$;
3  $Degree \leftarrow 1$;
4  while $Degree < MaxDegree$ do
5      $t \leftarrow Degree/MaxDegree$;
6      $S' \leftarrow$ select at random a solution in $\mathcal{V}_t(S)$;
7      $S \leftarrow$ OptimizedGreedy($S'$);
8      if $c(S') < c(S_{best})$ then
9          $S_{best} \leftarrow S'$;
10         $Degree \leftarrow 1$;
11     else
12         $Degree \leftarrow Degree + 1$
13 return $S_{best}$
```

for optimization. In Sect. 5, $MaxDegree = 16$. The algorithm VNSGreedy($MaxDegree$) has a complexity of $O(MaxDegreeN^2 \log N)$ (*cf.* Algorithm 3).

4.3 Summary

A greedy optimization of the descriptive criterion quickly evaluates many sets of prototypes by exploiting the additivity of the criterion. It is appealing to apply this algorithm repeatedly and we embed the heuristic in the framework of the Variable Neighborhood Search metaheuristic. The overall algorithm VNSGreedy($MaxDegree$) has a complexity of $O(MaxDegreeN^2 \log N)$. It is controlled by a single parameter $MaxDegree \in \mathbb{N}^*$ which represents the allowed training time. The method to optimize the MAP criterion with the VNS-Greedy heuristic will be referred to as Eva(*MaxDegree*) (*Eva* stands for supervised and nonparametric EVAluation of sets of instances) in the experiments.

5 Benchmarking Eva

In this section we comparatively evaluate several instance selection algorithms using four performance criteria. The experiment consists in applying the algorithms to datasets from the UCI Machine Learning Database Repository (Asuncion and Newman 2007) and measuring the size, the predictive accuracy and the reliability of the solution as well as the running time for each algorithm.

5.1 Description of the experiment

We compare our method Eva, the global method Explore and the local method ICF (*cf.* Sect. 2.3). The algorithm performing no selection is considered as well. The parameter *MaxDegree* of the VNSGreedy algorithm is set to 16. The parameters of the other methods are set as specified by their respective authors: the number of neighbors for the ENN rule (*i.e.* the preprocessing step of the ICF algorithm) is set to 3 and the number of mutations is set to 1000 for the Explore algorithm.

We select 23 datasets (*cf.* Table 1) from the UCI Machine Learning Database Repository (Asuncion and Newman 2007). The experiment carried out in this section aims at evaluating the methods through several indicators. The compression rate (*i.e.* the ratio of the number of prototypes to the number of instances) evaluates the capacity of a selection method to synthesize the dataset. The training time evaluates the effective duration of the selection process. The test accuracy evaluates the predictive performance of the selected set of prototypes on new instances, that is instances not participating to the selection of the prototypes. Two classification schemes are considered: the VBR scheme and the 1-NN rule (*cf.* Sect. 1.4). A discussion of this choice, based on some complementary experiments, is given in Appendix A. In order to test whether the classification is reliable, we introduce a *robustness* measure which is the ratio of the test accuracy to the training accuracy. A value of 100 percent means that the generalization ability of the classifier is high.

Every quantity is estimated using a stratified ten-fold cross-validation. A Student's t-test at the 5% confidence level is performed to determine whether the differences of predictive performance are significant.

We have to define a distance metric. We use the L_1 metric for the numerical features, the Hamming metric for the categorical features, and we sum the contributions. We choose to work as little as possible on the representation and on the definition of the distance metric in order not to bias our conclusions. We carry out some experiments in Appendix B in order to illustrate this point.

5.2 Results

According to Table 2, the global methods Eva and Explore perform better compression than the local one, retaining 1.5% (Eva) and 2.7% (Explore) of the instances on average against 18% (ICF). ICF is a local method that selects too many prototypes and cannot eradicate redundancy. Global methods evaluate the information that prototypes bring as a whole, which makes them able to avoid redundancy.

Table 1 Description of the datatsets

Datasets	Size	Number of features	Number of classes	Majority prediction
Iris	150	4	3	0.33
Wine	178	13	3	0.40
Sonar	208	60	2	0.53
Glass	214	9	6	0.36
Heart	270	13	2	0.56
Bupa	345	6	2	0.58
Ionosphere	351	33	2	0.64
Crx	690	15	2	0.56
Breast	699	9	2	0.66
Pima	768	8	2	0.65
Vehicle	846	18	4	0.26
Led	1000	7	10	0.11
Yeast	1484	8	10	0.31
Segmentation	2310	19	7	0.14
Abalone	4177	8	28	0.16
Spam	4307	57	2	0.65
Waveform	5000	21	3	0.34
WaveformNoise	5000	40	3	0.34
OpticalDigits	5620	64	10	0.10
Satimage	6435	36	6	0.24
Thyroid	7200	21	3	0.92
PenDigits	7494	16	10	0.10
Led17	10000	24	10	0.11

Comparing the two leading methods, Eva selects significantly fewer prototypes (15 wins and 1 loss). This results from the fact that Eva selects a prototype according to a fine-grained knowledge: the Eva criterion given by Eq. 14 takes into account the distribution of every class in each cell while the Explore criterion given by Eq. 2 only makes a binary distinction between the class of the prototype and the other classes. Furthermore, as shown in the next section, the VNSGreedy heuristic performs a better exploration of the search space.

The number of selected prototypes is closely related to the reliability of the method. If the number of prototypes decreases, the number of instances supporting each prototype increases and the labeling of any new instance should thus be more reliable. This fact is illustrated by the results in Table 3: the more reductions a method makes, the more reliable it is. Eva produces the most reliable predictions (97.1% on average).

Eva and Explore are reliable and their criteria are nonparametric. In real studies, the dataset is often divided in three subsets: one for building the models (the training set), one for adjusting the parameters independently (the validation set) and one for evaluating independently the accuracy of the final models (the test set). The use of the balanced criteria, as they are nonparametric, makes validation sets useless. The model thus benefits from the use of a bigger training set.

By reducing the number of prototypes (*cf.* Table 4), one could wonder whether such a reduction sacrifices predictive accuracy. The local method ICF(3) loses respectively 2.5% (for the VBR scheme) and 2.6% (for the 1-NN scheme) of accuracy on average when compared

Table 2 Compression rate of Eva(16), Explore(1000), ICF(3) and the Lazy algorithm (performing no selection), estimated with a stratified 10-fold cross-validation. The number of significant Wins/Losses of Eva is reported

	Eva	Explore	ICF	Lazy
Iris	2.3	3.1	40.2	100
Wine	2.2	2.7	22.7	100
Sonar	1.6	2.1	28.3	100
Glass	2.8	5.0	32.9	100
Heart	1.0	1.2	22.1	100
Bupa	0.8	1.0	17.5	100
Ionosphere	2.0	3.4	8.1	100
Crx	0.4	0.6	20.4	100
Breast	0.7	1.0	4.2	100
Pima	0.5	0.6	13.9	100
Vehicle	1.6	3.8	24.6	100
LED	12.0	11.6	37.3	100
Yeast	0.7	4.7	16.8	100
Segmentation	1.0	2.8	16.7	100
Abalone	0.1	1.9	13.9	100
Spam	0.1	0.1	18.6	100
Waveform	0.3	1.9	12.9	100
WaveformNoise	0.4	2.4	12.1	100
OpticalDigits	0.7	3.2	8.1	100
Satimage	0.5	1.7	8.4	100
Thyroid	0.0	0.1	3.5	100
PenDigits	0.6	2.0	6.5	100
LED17	1.0	5.3	24.7	100
Mean	1.5	2.7	18.0	100
W/L		15/1	23/0	23/0

Table 3 Mean of the robustness (in percent) of the VBR scheme and the 1-NN scheme, and number of significant Wins/Losses of Eva. The VBR scheme and the 1-NN scheme lead to the same decisions for the Lazy algorithm. The 1-NN scheme is meaningless for Eva

	Eva	Explore		ICF		Lazy
		NN	VBR	NN	VBR	
Mean	97.1	92.8	92.7	89.7	88.9	77.9
W/L		11/0	10/0	14/0	14/0	18/0

with the Lazy algorithm. As illustrated by the experiments in Sect. 7, the local methods rely on the hypothesis that the classes are separated. This hypothesis does not hold for every real dataset and such methods behave poorly when it is wrong.

With the selected datasets, we cannot conclude that Eva and Explore preserve predictive accuracy as they underperform the Lazy algorithm by 1.8% and 1.2% respectively. But it is very difficult to draw conclusions from the study of predictive accuracy when the standard deviation (not reported here) of 10 results out of 24 is above 6% for every tested method.

Table 4 Prediction accuracy of Eva(16), Explore(1000), ICF(3) and Lazy (performing no selection), according to the VBR scheme and the 1-NN scheme, estimated using a stratified 10-fold cross-validation. The number of significant Wins/Losses of Eva is reported. The VBR scheme and the 1-NN scheme lead to the same decisions for the Lazy algorithm. The 1-NN scheme is meaningless for Eva

	Eva	Explore		ICF		Lazy
		NN	VBR	NN	VBR	
Iris	96.0	96.0	96.0	92.0	92.0	95.3
Wine	88.9	81.0	81.0	74.2	74.8	83.2
Sonar	69.2	63.5	63.5	77.8	76.9	85.1
Glass	60.3	61.2	61.2	65.9	65.4	73.8
Heart	70.4	70.0	70.0	64.8	65.9	64.8
Bupa	67.5	62.4	62.4	62.3	61.2	60.6
Ionosphere	88.6	88.0	88.0	87.2	86.3	90.9
Crx	75.4	73.5	73.5	70.1	67.7	68.3
Breast	97.1	96.1	96.1	95.6	95.9	96.3
Pima	73.4	75.0	75.0	69.3	69.9	68.8
Vehicle	60.3	61.5	61.1	66.8	65.9	67.6
LED	56.4	66.9	66.3	58.3	58.5	72.6
Yeast	53.4	56.6	56.7	52.3	53.2	53.8
Segmentation	87.3	91.9	91.9	94.7	94.8	97.2
Abalone	24.9	25.6	25.9	23.0	24.4	20.8
Spam	83.8	81.2	81.2	83.0	83.0	85.2
Waveform	79.9	82.3	82.3	75.4	76.4	77.1
WaveformNoise	79.1	80.4	80.4	72.9	72.8	75.6
OpticalDigits	93.3	96.2	96.2	96.0	95.9	98.5
Satimage	87.0	89.2	89.2	87.2	88.2	90.7
Thyroid	93.2	93.6	93.6	91.6	93.8	93.2
PenDigits	95.6	98.0	98.0	98.1	98.0	99.5
LED17	48.7	53.8	53.8	53.7	51.4	52.0
Mean	75.2	75.8	75.8	74.4	74.5	77.0
W/L		3/8	3/8	8/6	7/6	6/9

Table 5 Mean of the training time (in second) of the tested methods and number of significant Win/Loss of Eva

	Eva	Explore	ICF	NN
Mean	108	5305	34	1
W/L		22/0	0/23	0/23

Some experiments are needed to better understand the predictive behavior of these methods. They are carried out in Sect. 7.

Table 5 completes the analysis to fully qualify the instance selection methods. The local one quickly builds a set of prototypes, but with a rather limited compression rate. In contrast, Eva and Explore build small and reliable sets of prototypes. But Explore is far more time demanding. Eva is in between, building even smaller and more reliable solutions than Explore, with only thrice the time needed by ICF.

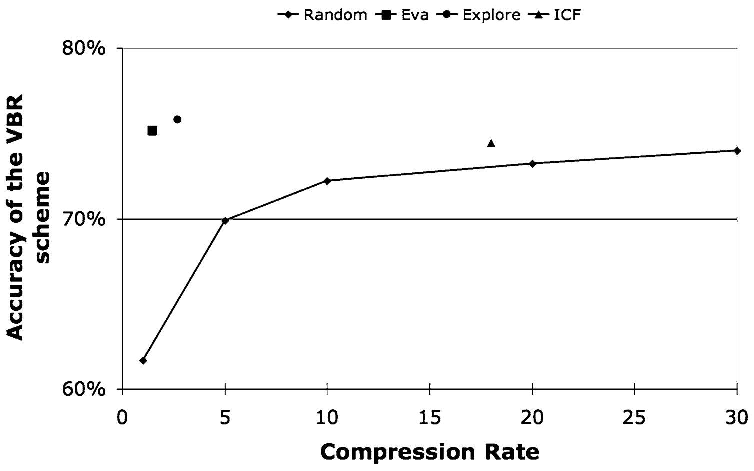


Fig. 3 Bicriteria evaluation of the methods, according to the compression rate and the predictive accuracy of the VBR scheme. The performance of random instance selection for various selection rates is reported as well

5.3 Summary

The study carried out on the UCI datasets shows that the local method runs fast but build sets of prototypes with limited compression. In contrast, the global methods Eva and Explore build small and reliable sets of prototypes. Among the two methods, Eva's compression rate is twice as high, with an increased reliability. Furthermore, Eva runs far faster than Explore. In Fig. 3, the methods are graphed according to their compression rate and the predictive accuracy of the VBR scheme.

6 Study of the VNSGreedy heuristic

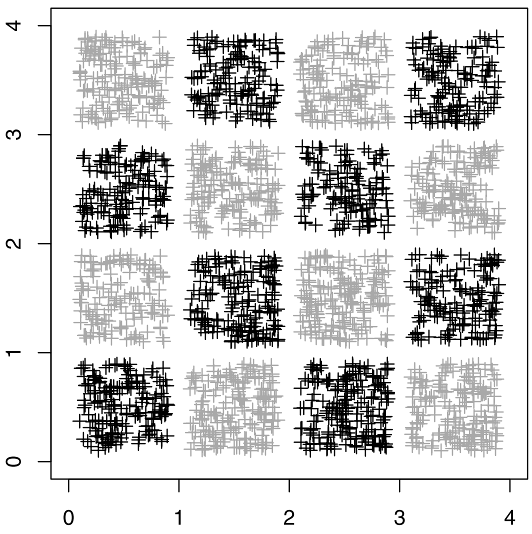
In this section, we illustrate the advantages of the new VNSGreedy heuristic on synthetic data. It is compared to the Explore heuristic. The performance is evaluated based on the training time, the value of Eva's criterion and the number of selected instances.

6.1 Description of the experiment

In this experiment, we uniformly generate a binary classification problem with 2500 instances lying on a 4 by 4 chessboard, as shown in Fig. 4. We also created a corrupted version of this dataset by uniformly mislabeling its instances with a probability of 0.2. Eva's criterion (i.e. Eq. 14) is optimized both with the VNSGreedy and Explore heuristics, with increasing values of their parameter. Furthermore, a set of 16 prototypes, each of which is the nearest instance of the center of a square, is evaluated. This set of prototypes is certainly a good solution and serves as a baseline.

The value of the criterion and the number of prototypes, evaluated using a stratified 5-fold cross-validation, are plotted against the training time for both datasets. The resulting curves are given in Fig. 5, for the "chessboard" dataset with no noise, and in Fig. 6, for the noisy "chessboard" dataset.

Fig. 4 The 4 by 4 "chessboard" dataset, with 2500 instances



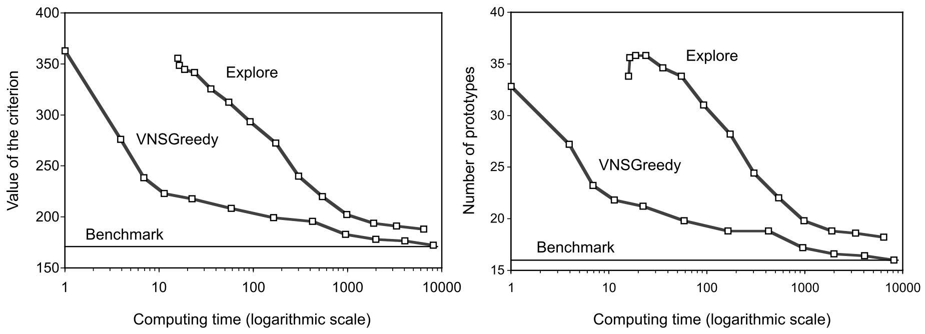


Fig. 5 Comparison of the VNSGreedy and the Explore heuristics on the chessboard dataset. The value of the criterion (14) and the number of prototypes are reported on the y-axis against the training time, reported on the x-axis

6.2 Results

In the noise-free case, the best thing to do is to build 16 pure areas. The baseline solution is thus a global optimum (evaluated at 172). The VNSGreedy algorithm outputs a first solution in 1 second (VNSGreedy(1), which consists of a single greedy optimization), while the Explore algorithm builds an equivalent solution in 16 seconds (Explore(0), which performs no mutations). The difference increases tremendously, as repeating the greedy optimizations allows to quickly find far better solutions: for a training time of 22 seconds, VNSGreedy(16) outputs a solution evaluated at 217, while Explore(2000) gives a value of 341. Finally, the VNSGreedy(2048) optimization outputs an optimal partition. In the same time, The Explore heuristic builds a partition with more than 18 groups.

This experiment on a noise-free dataset shows that the VNSGreedy heuristic always produces better solutions than Explore for a given training time or, in other words, equivalent solutions far more quickly.

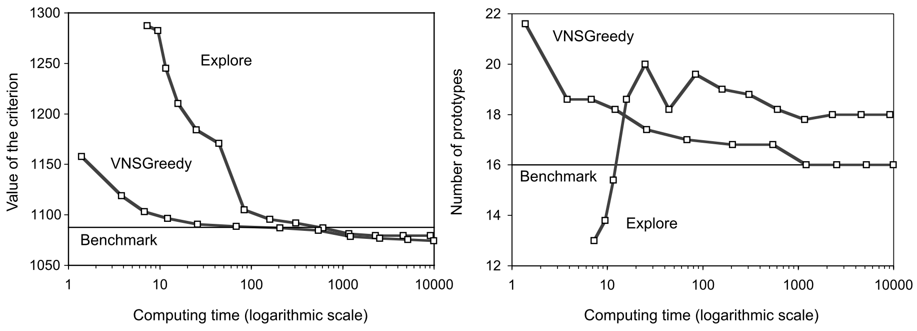


Fig. 6 Comparison of the VNSGreedy and the Explore heuristics on the noisy chessboard dataset. The value of the criterion (14) and the number of prototypes are reported on the y-axis against the training time, reported on the x-axis

In the noisy case, the benchmark solution (evaluated at 1088) is not necessarily a global optimum, as noisy instances can create random patterns. The introduction of noise also lessens the relative differences. However, the VNSGreedy heuristic still builds better solutions: for a given training time, VNSGreedy always outputs a solution with a lower value of the criterion. Furthermore, the Explore heuristic faces the challenge of escaping from local optima with more difficulties. The VNSGreedy(1) algorithm finds a first local optimum evaluated at 1157 while the Explore(0) algorithm falls into quite a bad local optimum, since this optimum is evaluated at 1287 and relies on 13 prototypes only. While the VNS metaheuristic can escape from local optima quite easily, the Explore heuristic applies a limiting strategy: by performing swaps, removals or adds one at a time, it takes about 100 seconds and 16,000 mutations for Explore to find a "good" optimum. Finally, this algorithm is asymptotically unable to escape from an optimum with 18 prototypes.

6.3 Summary

These experiments' results distinguish the behavior of the Explore heuristic and the new VNSGreedy heuristic. First, VNSGreedy(1) finds a solution far more quickly than Explore(0). Furthermore, the solution given by Explore(0) can be quite a bad local optimum. Its strategy for escaping from such an optimum is rather limited and then requires significantly more time to find a good optimum. The VNSGreedy optimization does not suffer from such a curse, as the adopted Variable Neighborhood Search metaheuristic strategy quickly enhances the solution.

7 Illustration of the behaviors

Each instance selection method is designed according to a particular aim: a local method relies on a specific and individual notion of interestingness of an instance and a global method optimizes a global criterion. Each interestingness measure and each global criterion carries its own semantic. We illustrate the behavior and the limits of each approach on two synthetic datasets.

7.1 First experiment

We uniformly generate a binary classification problem with 800 points lying inside the unit square. The distribution of the classes is $(80\%, 20\%)$ in the upper-right and lower-left corners, whereas it is $(0\%, 100\%)$ in the upper-left and lower-right corners. The dataset as well as the reduced sets built by ICF(3), Explore(1000) and Eva(16) are plotted in Fig. 7.

The ICF(3) algorithm applies the ENN rule as a preprocessing step. The ENN rule works as a noise filtering rule: instances misclassified by their L nearest neighbors are considered to be mislabeled and dropped. The parameter L corresponds to the decision level for which a pattern formed with at least L instances is considered significant. In this experiment $L = 3$, and as soon as there are at least 3 instances sufficiently close to each other and pertaining to the same class, those instances are not removed. Furthermore, instances with a different label next to those significant patterns are considered to be misclassified and dropped, giving more importance to such patterns. Tuning the parameter L is thus a delicate task.

The second step of the ICF algorithm removes central instances of pure areas: the bigger they are, the bigger the compression will be. Furthermore, as soon as a significant pattern is detected by the ENN rule, the method keeps the instances pertaining to a different class next to this pattern. This limits the influence of such a pattern but limits the compression as well.

Explore is based on a global evaluation resulting from the application of the MDL principle. This evaluation balances the predictive accuracy and the number of prototypes needed

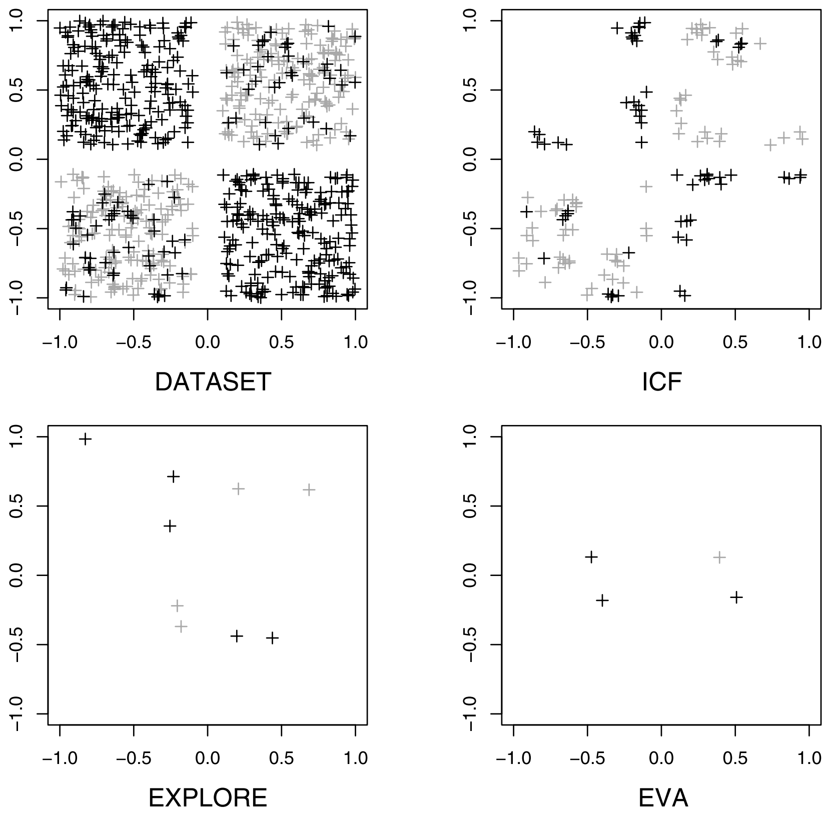


Fig. 7 Behaviors of different selection schemes on a synthetic dataset

to obtain this accuracy. The experiment illustrates that this method does not focus on small patterns: a small decrease of the error rate implies a great increase of the number of prototypes to keep and the small noisy patterns are removed. However, the optimization heuristic performs swaps at random and is not able to escape from a local optimum. Explore thus selects a bit too many instances, as it is the case in this experiment.

Eva aims to discriminate conditional distributions between cells. The method is designed for detecting different probabilistic behaviors of the target feature and this is exactly what it does here: Eva selects four instances as there are four areas (two pure, two slightly mixed).

7.2 Second experiment

Now, we illustrate the limits of the classical methods. We uniformly generate a binary classification problem with 2000 points lying inside the unit square. The distribution of the classes is (90%, 10%) in the upper-right and lower-left corners, whereas it is (60%, 40%) in the upper-left and lower-right corners. Here, the majority class is the same everywhere. The dataset as well as the reduced sets built by the ICF(3) algorithm, Explore(1000) and Eva(16) are plotted in Fig. 8.

This experiment illustrates the limit of the local method ICF: whereas the effects of the choice of L are well understood (*i.e.* it controls the size of the significant patterns), the use of 'significant pattern' is questionable. Indeed, such methods make the implicit assumption

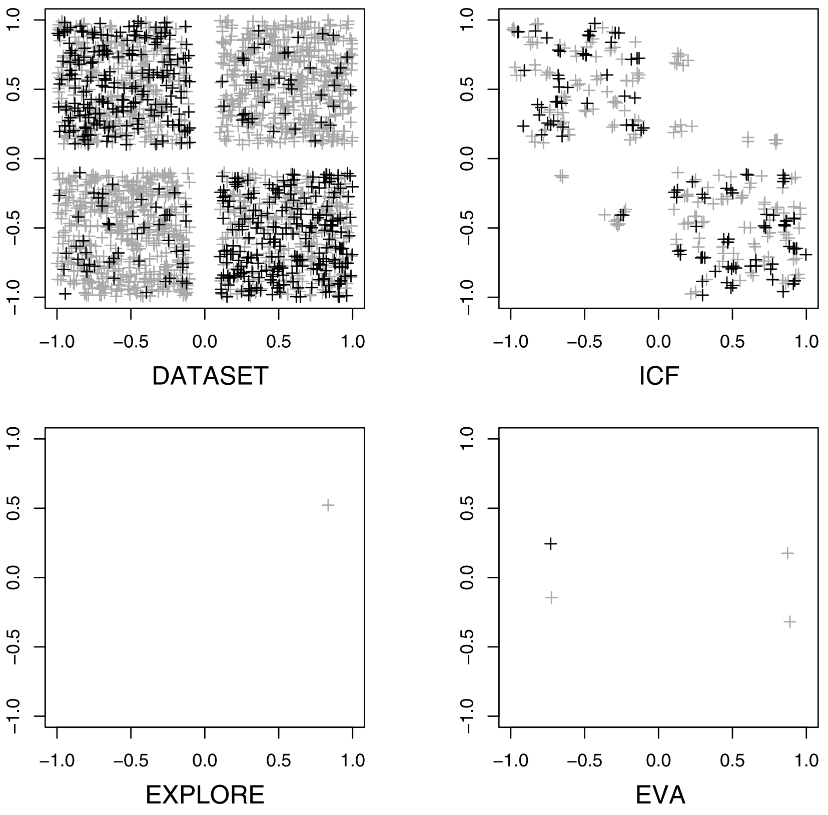


Fig. 8 Limits of different selection schemes on a synthetic dataset

that the instances form pure areas and that small areas are just some noise. If this assumption is true for some problems, it is not for others. The studied dataset is an example for which this assumption is false.

Intrinsically, this dataset illustrates the difference between evaluating classifiers (this is what Explore does) and evaluating conditional probabilities (this is what Eva does). Using a classification perspective, as the majority class is the same everywhere and as the minority class is spread out over the whole space, it is hopeless to find pure areas. Explore selects one prototype and builds the majority classifier. This method thus succeeds where previous ones fail: in the classification sense, there is nothing to extract from these data.

However, from a probabilistic point of view, the dataset brings information, as the target feature exhibits four different probabilistic behaviors. Eva, which aims to discriminate such behaviors, selects four prototypes. The fact that the upper-left prototype is of the minority class is not significant: according to the VBR classification scheme, the distribution of the label within the associated Voronoi cell is the only thing that matters.

7.3 Summary

As illustrated by the two experiments, the methods fall into two groups. The local methods rely on individual definitions of interestingness. However, these definitions make the implicit assumption that there are only pure areas in the dataset, which is not true in many practical cases. The global methods, when the evaluation is regularized, do not make this assumption. Further, Explore evaluates classifiers while Eva focuses on conditional probabilities, which are finer models. This is a key feature when the class of interest is not the majority one (consider the case of fraud detection or characterization of customer attrition, for example). Furthermore, the choice of an optimization heuristic for global evaluation is important as some heuristics have more difficulty to escape from local optima than others.

8 Conclusion

In this article, we have presented Eva, a new instance selection algorithm. It is designed for a relabeling classification scheme. The resulting decision rule is nonparametric and does not significantly sacrifice predictive accuracy when compared to the nearest neighbors rule. Eva outputs smaller and more reliable sets of instances within a competitive amount of time, outperforming alternative selection algorithms.

Eva relies on a maximum a posteriori evaluation of the conditional probabilities in each cell of the Voronoi partition associated with a set of instances. The evaluation is global and nonparametric. By using a probability distribution criterion (rather than a classification criterion), Eva is able to automatically tackle any kind of classification problem: from problems with separated classes to problems where the class of interest is the minority one. In order to make the most of the evaluation we have proposed a new and efficient optimization heuristic: VNSGreedy. The algorithm repeatedly applies a bottom-up greedy optimization according to a Variable Neighborhood Search meta-heuristic.

The next step consists in performing feature selection. The set of models must be extended to take into account subsets of features, the prior of the maximum a posteriori criterion must be extended to deal with the feature selection part of the models and the optimization heuristic must be extended to efficiently browse through the extended set of models. This is left as future work.

Appendix A: Influence of the classifying scheme

We carry out complementary experiments in order to compare the VBR and the K-NN classification schemes. The first experiment is similar to that in Sect. 5, the datasets and their stratified division into 10 folds remain identical. Selection is performed at random, with various selection rates. The predictive accuracy of the VBR and NN schemes are plotted in Fig. 9.

When the selection rate is low (around 1%), the VBR scheme appears to be the better choice (62% of predictive accuracy) while the 3-NN scheme is the worst (52% of predictive accuracy). Without relabeling, the prototype carries its own single label and, when the selection rate decreases, the classification is more and more blind. The classification according to the 3-NN scheme suffers even more from such a curse. In contrast, the relabeling tool used by the VBR scheme allows to take into account the whole distribution of the labels in the Voronoi cell of a prototype, making the classification more informed.

There is still some work remaining to select the right instances without decreasing accuracy, as the predictive accuracy of the Lazy algorithm, which performs no selection, is 77%: the more selective you are, the more difficult it is to retain a high predictive accuracy.

Appendix B: Influence of the representation

We carry out a complementary experiment in order to compare representations. The experiment is identical to the one in Sect. 5: the datasets, their stratified division into 10 folds, the methods and their parameterization. The only difference is in the representation: while the L_1 metric is still applied, the numerical features are first centered and reduced (the mean is subtracted and the result is divided by the standard deviation). The metric is still the Hamming distance for categorical features. Table 6 gives the prediction accuracy of the selection algorithms.

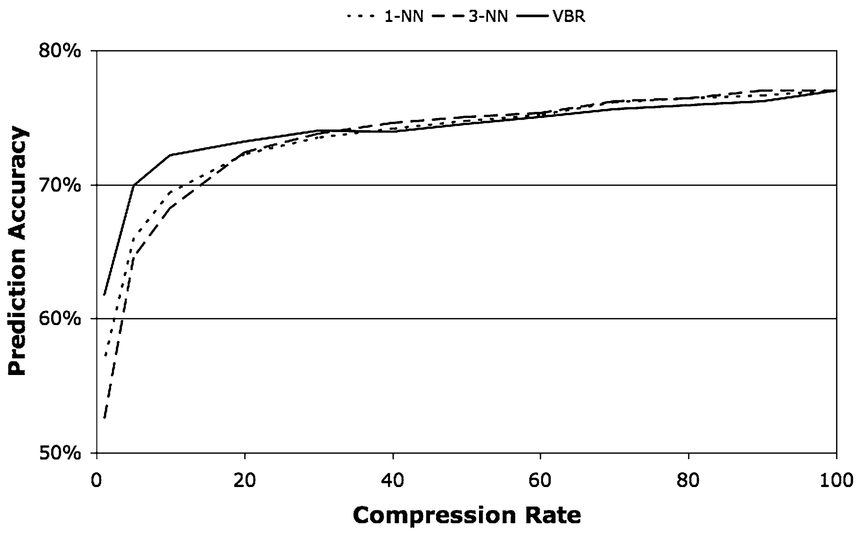


Fig. 9 Comparison of the predictive accuracy of the 1-NN, the 3-NN classification schemes and the Voronoi-based relabeling scheme. Instance selection is performed at random with varying selection rates

Table 6 Prediction accuracy of Eva(16), Explore(1000), ICF(3) and the Lazy algorithm (performing no selection), estimated using a stratified 10-fold cross-validation. The 23 datasets of the UCI are considered. The number of significant Wins/Losses of Eva is reported. All numerical features are centered and reduced before the application of the L_1 metric

	Eva	Explore			ICF			Lazy	
		1-NN	3-NN	VBR	1-NN	3-NN	VBR	1-NN	3-NN
Mean	77.7	79.0	68.7	79.0	76.5	76.8	77.1	79.1	79.5
W/L		1/10	11/6	1/10	7/6	4/5	5/5	6/8	3/9

Table 7 Prediction accuracy of Eva(16), Explore(1000), ICF(3) and the Lazy algorithm (performing no selection) for the Wine dataset, estimated using a stratified 10-fold cross-validation. All numerical features are centered and reduced before the application of the L_1 metric

Normalization	Eva	Explore			ICF			Lazy	
		1-NN	3-NN	VBR	1-NN	3-NN	VBR	1-NN	3-NN
without	88.9	81.0	48.4	81.0	74.2	71.9	74.8	83.2	77.5
with	96.6	95.5	63.8	95.5	92.1	94.3	92.1	97.2	97.2

Table 8 Means of the compression rate, the robustness and the training time of Eva(16), Explore(1000), ICF(3) and the Lazy algorithm (performing no selection) over 23 datasets, estimated using a stratified 10-fold cross-validation

	Eva	Explore (NN)	ICF (NN)	Lazy
compression rate	1.5	2.9	15.4	100
Robustness	96.7	93.3	90.9	79.9
training time	104	6904	35	1

The normalization of the features increases the predictive accuracy of every method. For example, the predictive accuracy of the lazy algorithm rises from 77.0% to 79.1% on average. This results from the reduction of the scaling differences between features with the normalization. Explore focuses on predictive accuracy and thus benefits more than Eva from this process.

There is a "representation masking effect". For example, let us have a closer look at the Wine dataset (*cf.* Table 7). Once the features have been normalized, the classes are easier to separate and any instance selection method is able to discriminate them. They obtain the same good prediction accuracy. We could be misled and draw the conclusion that selection algorithms are equivalent, while it is the representation that does the job. However, according to Table 8, the training time, the compression rate and the robustness of the selection process are not affected by such masking effect, strengthening the importance of a multi-criteria analysis.

Some additional experiments show that there is a "metric masking effect" as well. If we apply the VDM metric for categorical features (Stanfill and Waltz 1986), and as the definition of such metric integrates the target feature, it is even more difficult to discriminate between the benefits of the metric and that of the selection algorithm. This is why we prefer to adopt the simplest representation and distance metric to get a clear comparison between instance selection algorithms.

References

Aha, D. W. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, *36*(2), 267–287.

Aha, D., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Asuncion, A., & Newman, D. (2007). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Berndt, D., & Clifford, J. (1996). *Finding patterns in time series: a dynamic programming approach* (Tech. rep.). Advances Knowledge Discovery Data Mining.

Bhattacharya, B. K., Mukherjee, K., & Toussaint, G. T. (2005). Geometric decision rules for instance-based learning problems. In S. K. Pal, S. Bandyopadhyay, & S. Biswas (Eds.), *Lecture notes in computer science: Vol. 3776*. *PReMI* (pp. 60–69). Berlin: Springer.

Brighton, H., & Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, *6*(2), 153–172.

Bunke, H. (2000). Recent developments in graph matching. In *ICPR* (pp. 2117–2124).

Cameron-Jones, R. (1995). Instance selection by encoding length heuristic with random mutation hill climbing. In *Proceedings of the eighth Australian joint conference on artificial intelligence* (pp. 99–106).

Chang, C. (1991). Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, *23*(11), 1179–1184.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, *13*, 21–27.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Berlin: Springer.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern classification*. New York: Wiley.

Ferrandiz, S., & Boullé, M. (2006). Supervised evaluation of Voronoi partitions. *Intelligent Data Analysis*, *10*(3), 269–284.

Fix, E., & Hodges, J. (1951). *Discriminatory analysis. Nonparametric discrimination: consistency properties* (Technical Report 4). Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, TX.

Gates, G. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, *18*(3), 431–433.

Grünwald, P., Myung, I., & Pitt, M. (2005). *Advances in minimum description length: theory and applications*. Cambridge: MIT Press.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hansen, P., & Mladenovic, N. (2001). Variable neighborhood search: principles and applications. *European Journal of Operational Research*, *130*, 449–467.

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, *14*, 515–516.

Jaromczyk, J., & Toussaint, G. (1992). Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, *80*(9), 1502–1517.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin: Springer.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*(8), 707–710.

Liu, H., & Motoda, H. (2001). *Instance selection and construction for data mining*. Dordrecht: Kluwer.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. Cam, & Neyman (Eds.), *Fifth Berkeley symposium on mathematical statistics and probability* (pp. 281–297).

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, *33*(3), 1065–1076.

Robert, C. (2001). *The Bayesian choice: from decision-theoretic motivations to computational implementation*. New York: Springer.

Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, *6*, 277–309.

Sanchez, J. S., Pla, F., & Ferri, F. J. (1997). Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, *18*(6), 507–513.

Scholkopf, B., & Smola, A. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge: MIT Press.

Sebban, M., Nock, R., & Lallich, S. (2002). Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problem. *Journal of Machine Learning Research*, *3*, 863–885.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, *29*, 1213–1228.

Toussaint, G. T., Bhattacharya, B., & Poulsen, R. (1985). The application of Voronoi diagrams to nonparametric decision rules. In *Computer science and statistics: the interface* (pp. 97–108).

Toussaint, G. T., & Poulsen, R. (1975). Some new algorithms and software implementation methods for pattern recognition research. In *Proceedings of the international computer software applications conference* (pp. 55–63).

Vapnik, V. (1996). *The nature of statistical learning theory*. New York: Springer.

Wettschereck, D., & Dietterich, T. (1995). An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*, *19*(1), 5–27.

Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, *2*, 408–421.

Wilson, D., & Martinez, T. (1997a). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, *6*(1), 1–34.

Wilson, D., & Martinez, T. (1997b). Instance pruning techniques. In D. Fisher (Ed.), *Proceedings of the 14th international conference on machine learning* (pp. 403–411). San Francisco: Morgan Kaufmann.

Wilson, D., & Martinez, T. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*(3), 257–286.