Analogy-Based Reasoning in Classifier Construction

Arkadiusz Wojna

Institute of Informatics, Warsaw University,
Banacha 2, 02-097, Warsaw, Poland
wojna@mimuw.edu.pl

Abstract. Analogy-based reasoning methods in machine learning make it possible to reason about properties of objects on the basis of similarities between objects. A specific similarity based method is the k nearest neighbors (k-nn) classification algorithm. In the k-nn algorithm, a decision about a new object x is inferred on the basis of a fixed number k of the objects most similar to x in a given set of examples. The primary contribution of the dissertation is the introduction of two new classification models based on the k-nn algorithm.

The first model is a hybrid combination of the k-nn algorithm with rule induction. The proposed combination uses minimal consistent rules defined by local reducts of a set of examples. To make this combination possible the model of minimal consistent rules is generalized to a metric-dependent form. An effective polynomial algorithm implementing the classification model based on minimal consistent rules has been proposed by Bazan. We modify this algorithm in such a way that after addition of the modified algorithm to the k-nn algorithm the increase of the computation time is inconsiderable. For some tested classification problems the combined model was significantly more accurate than the classical k-nn classification algorithm.

For many real-life problems it is impossible to induce relevant global mathematical models from available sets of examples. The second model proposed in the dissertation is a method for dealing with such sets based on locally induced metrics. This method adapts the notion of similarity to the properties of a given test object. It makes it possible to select the correct decision in specific fragments of the space of objects. The method with local metrics improved significantly the classification accuracy of methods with global models in the hardest tested problems.

The important issues of quality and efficiency of the k-nn based methods are a similarity measure and the performance time in searching for the most similar objects in a given set of examples, respectively. In this dissertation both issues are studied in detail and some significant improvements are proposed for the similarity measures and for the search methods found in the literature.

Keywords: analogy-based reasoning, case-based reasoning, k nearest neighbors, similarity measure, distance based indexing, hybrid decision system, local metric induction.

1 Introduction

Decision-making as a human activity is often performed on different levels of abstraction. It includes both simple everyday decisions, such as selection of products while shopping, choice of itinerary to a workplace, and more compound decisions, e.g., in marking a student's work or in investments. Decisions are always made in the context of a current situation (i.e., the current state of the world) on the basis of the knowledge and experience acquired in the past. Computers support decision making. Several research directions have been developed to support computer-aided decision making. Among them are decision and game theory [57, 81], operational research [10], planning [28], control theory [67, 87], and machine learning [61]. The development of these directions has led to different methods of knowledge representation and reasoning about the real world for solving decision problems.

Decision-making is based on reasoning. There are different formal reasoning systems used by computers. Deductive reasoning [5] is based on the assumption that knowledge is represented and extended within a deductive system. This approach is very general and it encompasses a wide range of problems. However, real-life problems are usually very complex, and depend on many factors, some of them quite unpredictable. Deductive reasoning does not allow for such uncertainty. Therefore in machine learning another approach, called inductive reasoning [33, 50, 59], is used. Decision systems that implement inductive reasoning are based on the assumption that knowledge about a decision problem is given in the form of a set of examplary objects with known decisions. This set is called a training set. In the learning phase the system constructs a data model on the basis of the training set and then uses the constructed model to reason about the decisions for new objects called test objects. The most popular computational models used in inductive reasoning are: neural networks [15], decision trees [65], rule based systems [60], rough sets [63], bayesian networks [45], and analogy-based systems [68]. Inductive reasoning applied to large knowledge bases of objects made it possible to develop decision support systems for many areas of human activity, e.g., image, sound and handwriting recognition, medical and industrial diagnostics, credit decision making, fraud detection. Besides such general methods there are many specific methods dedicated to particular applications.

The goal of this dissertation is to present and analyze machine learning methods derived from the analogy-based reasoning paradigm [68], in particular, from case-based reasoning [3, 52]. Analogy-based reasoning reflects natural human reasoning that is based on the ability to associate concepts and facts by analogy. As in other inductive methods, we assume in case-based reasoning that a training set is given and reasoning about a new object is based on similar (analogous) objects from the training set.

Selection of a similarity measure among objects is an important component of this approach, which strongly affects the quality of reasoning. To construct a similarity measure and to compare objects we need to assume a certain fixed structure of objects. Most of data are collected in relational form: the objects are described by vectors of attribute values. Therefore, in the dissertation we assume this original structure of data. Numerous different metrics are used for such data [1, 14, 22, 51, 56, 77, 84, 88]. To construct such a metric one can use both general mathematical properties of the domains of attribute values and specific information encoded in the training data.

Case-based reasoning is more time-consuming than other inductive methods. However, the advance of hardware technology and the development of indexing methods for training examples [11, 29, 35, 43, 46, 62, 66, 78, 82] have made possible the application of case-based reasoning to real-life problems.

1.1 Results Presented in This Thesis

The research was conducted in two parallel directions. The first direction was based on the elaboration of reasoning methods and theoretical analysis of their quality and computational complexity. The second direction was focused on the implementation of the elaborated methods, and on experiments on real data followed by an analysis of experimental results. The quality of the methods developed was tested on data sets from the Internet data repository of the University of California at Irvine [16].

One of the widely used methods of case-based reasoning is the k nearest neighbors (k-nn) method [4, 23, 26, 31]. In the k-nn method the decision for a new object x is inferred from a fixed number k of the nearest neighbors of x in a training set. In the dissertation we present the following new methods and results related to the k-nn method:

1. A new metric for numerical attributes, called the Density Based Value Difference Metric (DBVDM) (Subsection 3.2),
2. An effective method for computing the distance between objects for the metrics WVDM [88] and DBVDM (Subsection 3.2),
3. Two attribute weighting algorithms (Subsections 3.4 and 3.5),
4. A new indexing structure and an effective searching method for the k nearest neighbors of a given test object (Section 4),
5. A classification model that combines the k-nn method with rule based filtering (Subsections 5.3 and 5.4),
6. The k-nn classification model based on locally induced metrics (Subsection 5.6).

Below we provide some detailed comments on the results of the dissertation.

Ad.(1). In case of the classical k-nn method is an important quality factor the selection of an appropriate similarity measure among objects[1, 2, 14, 22, 25, 51], [56, 77, 84, 85, 89, 88]. To define such a metric, in the first place, some general mathematical properties of the domains of attribute values can be used. The fundamental relation for comparing attribute values is the equality relation: for any pair of attribute values one can check if they are equal or not. Other relations on attribute values depend on the attribute type. In typical relational databases two types of attributes occur. Nominal attributes (e.g., color, shape, sex) are the most general. The values of such attributes can only be compared by the equality relation. The values of numerical attributes (e.g., size, age, temperature) are represented by real numbers. Numerical attributes provide more information about relations between values than nominal attributes, e.g., due to their linearly ordered structure and the existence of a measure of distance between values. The examples of metrics using only general relations on attribute values are the Hamming distance for nominal attributes and the l_p or the χ-square distance for numerical attributes.

However, in decision making such general relations on attribute values are not relevant, as they do not provide information about the relation between the values of attributes and the decision. Hence, an additional source of information, i.e., a training set, is used to construct a metric. By contrast to the properties of general relations on attribute values, this information depends on the problem to be solved and it helps to recognize which attributes and which of their properties are important in decision making for this particular problem. An example of such a data-dependent metric is provided by the Value Difference Metric (VDM) for nominal attributes. The VDM distance between two nominal values is defined on the basis of the distance between decision distributions for these two values in a given training set [77, 22]. Wilson and Martinez [88] proposed analogous metrics for numerical attributes: the Interpolated Value Difference Metric (IVDM) and the Windowed Value Difference Metric (WVDM). By analogy to VDM both metrics assign a decision distribution to each numerical value. To define such an assignment, for both metrics the objects whose values fall into a certain interval surrounding this value are used. The width of this interval is constant: it does not depend on the value. In many data sets the density of numerical values depends strongly on the values, and the constant width of the interval to be sampled can lead to the situation where for some values the sample obtained is not representative: it can contain either too few or too many objects.

In the dissertation we introduce the Density Based Value Difference Metric (DBVDM). In DBVDM the width of the interval to be sampled depends on the density of attribute values in a given training set. In this way we avoid the problem of having either too few or too many examples in the sample.

Ad.(2). The time required to compute the decision distribution for each numerical value by means of WVDM or DBVDM is linear with respect to the training set size. Hence, it is impractical to perform such a computation every time one needs the distance between two objects. In the dissertation we show that the decision distributions for all the values of a numerical attribute can be computed in total time $O(n \log n)$ (where n is the size of the given training set). This allows to compute the distance between two objects in logarithmic or even in constant time after preliminary conversion of the training set. This acceleration is indispensable if WVDM or DBVDM is to be applied to real-life data.

Ad.(3). Usually in real-life problems there are some factors that make attributes unequally important in decision making. The correlation of some attributes with the decision is stronger. Moreover, some attribute values are influenced by noise, which makes them less trustworthy than exact values of other attributes. Therefore, to ensure good similarity measure quality it is important to use attribute weights in its construction. Much research has been done on the development of algorithms for on-line optimization of attribute weights (i.e., training examples are processed sequentially and weights are modified after each example) [2, 48, 51, 69]. However, for real-life data sets the k-nn classification requires an advanced indexing method. We discuss this in more detail in the dissertation. In this case on-line algorithms are ineffective: indexing must be performed each time attribute weights are modified, i.e., after each example. Another disadvantage of on-line algorithms is that they are sensitive to the order of training examples.

Attribute weighting algorithms, used in practice, are batch algorithms with a small number of iterations, i.e., the algorithms that modify attribute weights only after having processed all the training examples. Lowe [56] and Wettschereck [84] have proposed such algorithms. Both algorithms use the conjugate gradient to optimize attribute weights, which means minimizing a certain error function based on the leave-one-out test on the training set. However, Lowe and Wettischereck's methods are applicable only to the specific weighted Euclidean metric.

In the dissertation we introduce two batch weighting algorithms assuming only that metrics are defined by a weighted linear combination of metrics for particular attributes. This assumption is less restrictive: attribute weighting can be thus applied to different metrics. The first algorithm proposed optimizes the distance to the objects classifying correctly in the leave-one-out test on the training set. The second algorithm optimizes classification accuracy in the leave-one-out test on the training set. We performed experiments consisting in the application of the proposed weighting methods to different types of metrics and in each case the weighting algorithms improved metric accuracy.

Ad.(4). Real-life data sets collected in electronic databases often consist of thousands or millions of records. To apply case-based queries to such large data tables some advanced metric-based indexing methods are required. These methods can be viewed as the extension of query methods expressed in the SQL language in case of relational databases where the role of similarity measure is taken over by indices and foreign keys, whereas similarity is measured by the distance between objects in an index and belonging the ones to the same set at grouping, respectively.

Metric-based indexing has attracted the interest of many researchers. Most of the methods developed minimize the number of I/O operations [9, 12, 13, 20, 43, 47], [55, 62, 66, 71, 83, 86]. However, the increase in RAM memory available in modern computers makes it possible to load and store quite large data sets in this fast-access memory and indexing methods based on this type of storage gain in importance. Efficiency of indexing methods of this type is determined by the average number of distance computations performed while searching a database for objects most similar to a query object. The first methods that reduces the number of distance computations have been proposed for the case of a vector space [11, 29]. They correspond to the specific Euclidean metric.

In the dissertation we consider different metrics defined both for numerical and nominal attributes and therefore we focus on more general indexing methods, such as BST [46], GHT [78], and GNAT [18]. GHT assumes that only a distance computing function is provided. BST and GNAT assume moreover that there is a procedure that computes center of an object set, which corresponds to computing the mean in a vector space. However, no assumptions about the properties of the center are used. In each of these two methods both the indexing and searching algorithms are correct for any definition of center. Such a definition affects only search efficiency.

In the most popular indexing scheme the indexing structure is constructed in the form of a tree. The construction uses the top-down strategy: in the beginning the whole training set is split into a number of smaller nodes and then each node obtained is recursively split into smaller nodes. Training objects are assigned to the leaves. All the three indexing methods (BST, GHT, and GNAT) follow this general scheme. One of the important components that affects the efficiency of such indexing trees is the node splitting procedure. BST, GHT, and GNAT use a single-step splitting procedure, i.e., the splitting algorithm selects criteria to distribute the objects from a parent node and then assigns the objects to child nodes according to these criteria. At each node this operation is performed once. In the dissertation we propose an indexing tree with an iterative k-means-like splitting procedure. Savaresi and Boley have shown that such a procedure has good theoretical splitting properties [70] and in the experiments we prove that the indexing tree with this iterative splitting procedure is more efficient than trees with a single-step procedure.

Searching in a tree-based indexing structure can be speeded up in the following way: the algorithm finds quickly the first candidates for the nearest neighbors and then it excludes branches that are recognized to contain no candidates closer than those previously found. Each of the three methods BST, GHT, and GNAT uses a different single mathematical criterion to exclude branches of the indexing tree. However, all the three criteria assume similar properties of the indexing tree. In the dissertation we propose a search algorithm that uses all the three criteria simultaneously. We show experimentally that for large data sets the combination of this new search algorithm with the iterative splitting based tree makes nearest neighbors searching up to several times more efficient than the methods BST, GHT, and GNAT. This new method allows us to apply the k-nn method to data with several hundred thousand training objects and for the largest tested data set it makes it possible to reduce the 1-nn search by 4000 times as compared with linear search.

Ad.(5). After defining a metric and choosing a method to speed up the search for similar objects, the last step is the selection of a classification model. The classical k-nn method finds a fixed number k of the nearest neighbors of a test object in the training set, assigns certain voting weights to these nearest neighbors and selects the decision with the greatest sum of voting weights.

The metric used to find the nearest neighbors is the same for each test object: it is induced globally from the training set. Real-life data are usually too complex to be accurately modeled by a global mathematical model. Therefore such a global metric can only be an approximation of similarity encoded in data and it can be inaccurate for specific objects. To ensure that the k nearest neighbors found for a test object are actually similar, a popular solution is to combine the k-nn method with another classification model.

A certain improvement in classification accuracy has been observed for models combining the k-nn approach with rule induction [25, 37, 54]. All these models use the approach typical for rule induction: they generate a certain set of rules a priori and then they apply these generated rules in the classification process. Computation of an appropriate set of rules is usually time-consuming: to select accurate rules algorithms need to evaluate certain qualitative measures for rules in relation to the training set.

In the dissertation we propose a classification model combining the k-nn with rule induction in such a way that after addition of the rule based component the increase of the performance time of the k-nn method is inconsiderable. The k-nn implements the lazy learning approach where computation is postponed till the moment of classification [6, 34]. We add rule induction to the k nearest neighbors in such a way that the combined model preserves lazy learning, i.e., rules are constructed in a lazy way at the moment of classification.

The combined model proposed in the dissertation is based on the set of all minimal consistent rules in the training set [74]. This set has good theoretical properties: it corresponds to the set of all the rules generated from all local reducts of the training set [94]. However, the number of all minimal consistent rules can be exponential with respect both to the number of attributes and to the training set size. Thus, it is practically impossible to generate them all. An effective lazy simulation of the classification based on the set of all minimal consistent rules for data with nominal attributes has been described by Bazan [6]. Instead of computing all minimal consistent rules a priori before classification the algorithm generates so called local rules at the moment of classification. Local rules have specific properties related to minimal consistent rules and, on the other hand, they can be effectively computed. This implies that classification based on the set of all minimal consistent rules can be simulated in polynomial time.

In the dissertation we introduce a metric-dependent generalization of the notions of minimal consistent rule and local rule. We prove that the model of rules assumed by Bazan [6] is a specific case of the proposed generalization where the metric is assumed to be the Hamming metric. We show that the generalized model has properties analogous to those of the original model: there is a relationship between generalized minimal consistent rules and generalized local rules that makes the application of Bazan's lazy algorithm to the generalized model possible.

The proposed metric-dependent generalization enables a combination of Bazan's lazy algorithm with the k-nn method. Using the properties of the generalized model we modify Bazan's algorithm in such a way that after addition of the modified algorithm to the k-nn the increase of the performance time is insignificant.

We show that the proposed rule-based extension of the k-nn is a sort of voting by the k nearest neighbors that can be naturally combined with any other voting system. It can be viewed as the rule based verification and selection of similar objects found by the k-nn classifier. The experiments performed show that the proposed rule-based voting gives the best classification accuracy when combined with a voting model where the nearest neighbors of a test object are assigned the inverse square distance weights. For some data sets the rule based extension added to the k-nn method decreases relatively the classification error by several tens of percent.

Ad.(6). The k-nn, other inductive learning methods and even hybrid combinations of these inductive methods are based on the induction of a mathematical model from training data and application of this model to reasoning about test objects. The induced data model remains invariant while reasoning about different test objects. For many real-life data it is impossible to induce relevant global models. This fact has been recently observed by researches in different areas, like data mining, statistics, multiagent systems [17, 75, 79]. The main reason is that phenomena described by real-life data are often too complex and we do not have sufficient knowledge in data to induce global models or a parameterized class of such models together with searching methods for the relevant global model in such a class.

In the dissertation we propose a method for dealing with such real-life data. The proposed method refers to another approach called transductive learning [79]. In this approach the classification algorithm uses the knowledge encoded in the training set, but it also uses knowledge about test objects in construction of classification models. This means that for different test objects different classification models are used. Application of transductive approach to problem solving is limited by longer performance time than in inductive learning, but the advance of hardware technology makes this approach applicable to real problems.

In the classical k-nn method the global, invariant model is the metric used to find the nearest neighbors of test objects. The metric definition is independent of the location of a test object, whereas the topology and the density of training objects in real data are usually not homogeneous. In the dissertation we propose a method for inducing a local metric for each test object and then this local metric is used to select the nearest neighbors. Local metric induction depends locally on the properties of the test object, therefore the notion of similarity can be adapted to these properties and the correct decision can be selected in specific distinctive fragments of the space of objects.

Such a local approach to the k-nn method has been already considered in literature [24, 32, 44]. However, all the methods described in literature are specific: they can be applied only to data from a vector space and they are based on local adaptation of a specific global metric in this space. In the dissertation we propose a method that requires a certain global metric but the global metric is used only for a preliminary selection of a set of training objects used to induce a local metric. This method is much more general: it makes the global metric and the local metric independent and it allows us to use any metric definition as a local metric.

In the experiments we show that the local metric induction method is helpful in the case of hard classification problems where the classification error of different global models remains high. For one of the tested data sets this method obtaines the classification accuracy that has never been reported before in literature.

Partial results from the dissertation have been published and presented at the international conferences RSCTC, ECML and ICDM [8, 39, 38, 76, 91] and in the journal Fundamenta Informaticae [40, 92]. The methods described have been implemented and they are available in the form of a software library and in the system RSES [8, 73].

1.2 Organization of the Thesis

Section 2 introduces the reader to the problem of learning from data and to the evaluation method of learning accuracy (Subsections 2.1 and 2.2). It describes the basic model of analogy-based learning, the k-nn (Subsections 2.3–2.5), and it presents the experimental methodology used in the dissertation (Subsections 2.6 and 2.7).

Section 3 introduces different metrics induced from training data. It starts with the definition of VDM for nominal attributes (Subsection 3.1). Then, it describes three extensions of the VDM metric for numerical attributes: IVDM, WVDM and DBVDM, and an effective algorithm for computing the distance between objects for these metrics (Subsection 3.2). Next, two attribute weighting algorithms are presented: an algorithm that optimizes distance and an algorithm that optimizes classification accuracy (Subsections 3.3–3.5). Finally, experiments comparing accuracy of the described metrics and weighting methods are presented (Subsections 3.6–3.9).

Section 4 describes the indexing tree with the iterative splitting procedure (Subsections 4.2–4.4), and the nearest neighbors search method with three combined pruning criteria (Subsections 4.5 and 4.6). Moreover, It presents an experimental comparison of this search method with other methods known from the literature (Subsections 4.7 and 4.8).

In Section 5, first we describe the algorithm that estimates automatically the optimal number of neighbors k in the k-nn classifier (Subsection 5.1). The rest of the section is dedicated to two new classification models that use previously described components: the metrics, indexing and the estimation of the optimal k. First, the metric-based extension of rule induction and the combination of a rule based classification model with the k nearest neighbors method is described and compared experimentally with other known methods (Subsections 5.3–5.5). Next, the model with local metric induction is presented and evaluated experimentally (Subsections 5.6 and 5.7).

2 Basic Notions

In this section, we define formally the problem of concept learning from examples.

2.1 Learning a Concept from Examples

We assume that the target concept is defined over a universe of objects U^∞. The concept to be learnt is represented by a decision function $dec : U^\infty \rightarrow V_{dec}$. In the thesis we consider the situation, when the decision is discrete and finite $V_{dec} = \{d_1, \ldots d_m\}$. The value $dec(x) \in V_{dec}$ for an object $x \in U^\infty$ represents the category of the concept that the object x belongs to.

In the thesis we investigate the problem of decision learning from a set of examples. We assume that the target decision function $dec : U^\infty \rightarrow V_{dec}$ is unknown. Instead of this there is a finite set of training examples $U \subseteq U^\infty$ provided, and the decision values $dec(x)$ are available for the objects $x \in U$ only. The task is to provide an algorithmic method that learns a function (hypothesis) $h : U^\infty \rightarrow V_{dec}$ approximating the real decision function dec given only this set of training examples U.

The objects from the universe U^∞ are real objects. In the dissertation we assume that they are described by a set of n attributes $A = \{a_1, \ldots, a_n\}$. Each real object $x \in U^\infty$ is represented by the object that is a vector of values $(x_1, \ldots, x_n)$. Each value x_i is the value of the attribute a_i on this real object x. Each attribute $a_i \in A$ has its domain of values V_i and for each object representation $(x_1, \ldots, x_n)$ the values of the attributes belong to the corresponding domains: $x_i \in V_i$ for all $1 \leq i \leq n$. In other words, the space of object representations is defined as the product $\mathbb{X} = V_1 \times \ldots \times V_n$. The type of an attribute a_i is either numerical, if its values are comparable and can be represented by numbers $V_i \subseteq \mathbb{R}$ (e.g., age, temperature, height), or nominal, if its values are incomparable, i.e., if there is no linear order on V_i (e.g., color, sex, shape).

It is easy to learn a function that assigns the appropriate decision for each object in a training set $x \in U$. However, in most of decision learning problems a training set U is only a small sample of possible objects that can occur in real application and it is important to learn a hypothesis h that recognizes correctly as many objects as possible. The most desirable situation is to learn the hypothesis that is accurately the target function: $h(x) = dec(x)$ for all $x \in U^\infty$. Therefore the quality of a learning method depends on its ability to generalize information from examples rather than on its accuracy on the training set.

The problem is that the target function dec is usually unknown and the information about this function dec is restricted only to a set of examples. In such a situation a widely used method to compare different learning algorithms is to divide a given set of objects U into a training part U_{trn} and a test part U_{tst}, next, to apply learning algorithms to the training part U_{trn}, and finally, to measure accuracy of the induced hypothesis on the test set U_{tst} using the proportion of the correctly classified objects to all objects in the test set [61]:

$$accuracy(h) = \frac{|\{x \in U_{tst} : \; h(x) = dec(x)\}|}{|U_{tst}|}.$$

2.2 Learning as Concept Approximation in Rough Set Theory

The information available about each training object $x \in U_{trn}$ is restricted to the vector of attribute values $(x_1, \ldots, x_n)$ and the decision value $dec(x)$. This defines the indiscernibility relation $IND = \{(x, x') : \forall a_i \in A\ x_i = x'_i\}$. The indiscernibility relation IND is an equivalence relation and defines a partition in the set of the training objects U_{trn}. The equivalence class of an object $x \in U_{trn}$ is defined by $IND(x) = \{x' : xINDx'\}$. Each equivalence class contains the objects that are indiscernible by the values of the attributes from the set A. The pair (U_{trn}, IND) is called an approximation space over the set U_{trn} [63, 64].

Each decision category $d_j \in V_{dec}$ is associated with its decision class in U_{trn}: $Class(d_j) = \{x \in U_{trn} : dec(x) = d_j\}$. The approximation space $AS = (U_{trn}, IND)$ defines the lower and upper approximation for each decision class:

$$LOWER_{AS}(Class(d_j)) = \{x \in U_{trn} : IND(x) \subseteq Class(d_j)\}$$
$$UPPER_{AS}(Class(d_j)) = \{x \in U_{trn} : IND(x) \cap Class(d_j) \neq \emptyset\}$$

The problem of concept learning can be described as searching for an extension (U^∞, IND^∞) of the approximation space (U_{trn}, IND), relevant for approximation of the target concept dec. In such an extension each new object $x \in U^\infty$ provides an information $(x_1, \ldots, x_n) \in \mathbb{X}$ with semantics $IND^\infty(x) \subseteq U^\infty$. By $\|(x_1, \ldots, x_n)\|_{U_{trn}}$ and $\|(x_1, \ldots, x_n)\|_{U^\infty}$ we denote the semantics of the pattern $(x_1, \ldots, x_n)$ in U_{trn} and U^∞, respectively. Moreover, $\|(x_1, \ldots, x_n)\|_{U_{trn}} = IND(x)$ and $\|(x_1, \ldots, x_n)\|_{U^\infty} = IND^\infty(x)$.

In order to define the lower and upper approximation of $Class(d_j) \subseteq U^\infty$ using IND^∞ one should estimate the relationships between $IND^\infty(x)$ and $Class(d_l)$ for $l = 1, \ldots, m$.

In the dissertation two methods are used.

In the first method we estimate the relationships between $IND^\infty(x)$ and $Class(d_l)$ by:

1. selecting from U_{trn} the set $NN(x, k)$ of k nearest neighbors of x by using a distance function (metric) defined on patterns;
2. using the relationships between $\|(y_1, \ldots, y_n)\|_{U_{trn}}$ and $Class(d_l) \cap U_{trn}$ for $y \in NN(x, k)$ and $l = 1, \ldots, m$ to estimate the relationship between IND^∞ (x) and $Class(d_j)$.

One can also use another method for estimating the relationship between $IND^\infty(x)$ and $Class(d_j)$. Observe that the patterns from $\{(y_1, \ldots, y_n) : y \in U_{trn}\}$ are not enough general for matching arbitrary objects from U^∞. Hence, first, using a distance function we generalize the patterns $(y_1, \ldots, y_n)$ for $y \in U_{trn}$ to patterns $pattern(y)$ that are combinations of so called generalized descriptors $a_i \in W$, where $W \subseteq V_i$, with the semantics $\|a_i \in W\|_{U_{trn}} = \{y \in U_{trn} : y_i \in W\}$. The generalization preserves the following constraint: if $\|(y_1, \ldots, y_n)\|_{U_{trn}} \subseteq Class(d_l)$ then $\|pattern(y)\|_{U_{trn}} \subseteq Class(d_l)$. For a given $x \in U^\infty$ we select all *pattern*(y) that are matching x and we use the relationships between their semantics and $Class(d_l)$ for $l = 1, \ldots, m$ to estimate the relationship between $IND^{\infty}(x)$ and $Class(d_j)$.

Since in the considered problem of concept learning the only information about new objects to be classified is the vector of attribute values $(x_1, \ldots, x_n) \in \mathbb{X}$ the objects with the same value vector are indiscernible. Therefore searching for a hypothesis $h : U^{\infty} \rightarrow V_{dec}$ approximating the real function $dec : U^{\infty} \rightarrow V_{dec}$ is restricted to searching for a hypothesis of the form $h : \mathbb{X} \rightarrow V_{dec}$. To this end the space of object representations $\mathbb{X}$ is called for short the space of objects and we consider the problem of learning a hypothesis using this restricted form $h : \mathbb{X} \rightarrow V_{dec}$.

2.3 Metric in the Space of Objects

We assume that in the space of objects $\mathbb{X}$ a distance function $\rho : \mathbb{X}^2 \rightarrow \mathbb{R}$ is defined. The distance function ρ is assumed to satisfy the axioms of a pseudometric, i.e., for any objects $x, y, z \in \mathbb{X}$:

1. $\rho(x, y) \geq 0$ (positivity),
2. $\rho(x, x) = 0$ (reflexivity),
3. $\rho(x, y) = \rho(y, x)$ (symmetry),
4. $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$ (triangular inequality).

The distance function ρ models the relation of similarity between objects. The properties of symmetry and triangular inequality are not necessary to model similarity but they are fundamental for the efficiency of the learning methods described in this thesis and for many other methods from the literature [9, 11, 12, 18, 19, 20, 29, 35, 36]. Sometimes the definition of a distance function satisfies the strict positivity: $x \neq y \Rightarrow \rho(x, y) > 0$. However, the strict positivity is not used by the distance based learning algorithms and a number of important distance measures like VDM [77] and the metrics proposed in this thesis do not satisfy this property.

In the l_p-norm based metric the distance between two objects $x=(x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ is defined by

$$\rho(x, y) = \left(\sum_{i=1}^{n} \rho_i(x_i, y_i)^p \right)^{\frac{1}{p}}$$

where the metrics ρ_i are the distance functions defined for particular attributes $a_i \in A$.

Aggarwal et al. [1] have examined the meaningfulness of the concept of similarity in high-dimensional real value spaces investigating the effectiveness of the l_p-norm based metric in dependence on the value of the parameter p. They proved the following result:

Theorem 1. *For the uniform distribution of 2 points x, y in the cube $(0,1)^n$ with the norm l_p ($p \geq 1$):*

$$\lim_{n\to\infty} E\left[\left(\frac{\max(\|x\|_p, \|y\|_p) - \min(\|x\|_p, \|y\|_p)}{\min(\|x\|_p, \|y\|_p)}\right)\sqrt{n}\right] = C\sqrt{\frac{1}{2p+1}}$$

where C is a positive constant and $\|\cdot\|_p$ denotes the standard norm in the space l_p.

It shows that the smaller p, the larger relative contrast is between the point closer to and the point farther from the beginning of the coordinate system. It indicates that the smaller p the more effective metric is induced from the l_p-norm. In the context of this result $p = 1$ is the optimal trade-off between the quality of the measure and its properties: $p = 1$ is the minimal index of the l_p-norm that preserves the triangular inequality. The fractional distance measures with $p < 1$ do not have this property.

On the basis of this result we assume the value $p = 1$ and in the thesis we explore the metrics that are defined as linear sum of metrics ρ_i for particular attributes $a_i \in A$:

$$\rho(x, y) = \sum_{i=1}^{n} \rho_i(x_i, y_i). \tag{1}$$

In the problem of learning from a set of examples U_{trn} the particular distance functions ρ_i are induced from a training set U_{trn}. It means that the metric definition depends on the provided examples and it is different for different data sets.

2.4 City-Block and Hamming Metric

In this subsection we introduce the definition of a basic metric that is widely used in the literature. This metric combines the city-block (Manhattan) distance for the values of numerical attributes and the Hamming distance for the values of nominal attributes.

The distance $\rho_i(x_i, y_i)$ between two values x_i, y_i of a numerical attribute a_i in the city-block distance is defined by

$$\rho_i(x_i, y_i) = |x_i - y_i|. \tag{2}$$

The scale of values for different domains of numerical attributes can be different. To make the distance measures for different numerical attributes equally significant it is better to use the normalized value difference. Two types of normalization are used. In the first one the difference is normalized with the range of the values of the attribute a_i

$$\rho_i(x_i, y_i) = \frac{|x_i - y_i|}{max_i - min_i}, \tag{3}$$

where $max_i = \max_{x \in U_{trn}} x_i$ and $min_i = \min_{x \in U_{trn}} x_i$ are the maximal and the minimal value of the attribute a_i in the training set U_{trn}. In the second type of normalization the value difference is normalized with the standard deviation of the values of the attribute a_i in the training set U_{trn}:

$$\rho_i(x_i, y_i) = \frac{|x_i - y_i|}{2\sigma_i}$$

where $\sigma_i = \sqrt{\frac{\sum_{x \in U_{trn}} (x_i - \mu_i)^2}{|U_{trn}|}}$ and $\mu_i = \frac{\sum_{x \in U_{trn}} x_i}{|U_{trn}|}$.

The distance $\rho_i(x_i, y_i)$ between two nominal values x_i, y_i in the Hamming distance is defined by the Kronecker delta:

$$\rho_i(x_i, y_i) = \begin{cases} 1 \text{ if } x_i \neq y_i \\ 0 \text{ if } x_i = y_i. \end{cases}$$

The combined city-block and Hamming metric sums the normalized value differences for numerical attributes and the values of Kronecker delta for nominal attributes. The normalization of numerical attributes with the range of values $max_i - min_i$ makes numerical and nominal attributes equally significant: the range of distances between values is $[0; 1]$ for each attribute. The only possible distance values for nominal attributes are the limiting values 0 and 1, whereas the normalized distance definition for numerical attributes can give any value between 0 and 1. It results from the type of an attribute: the domain of a nominal attribute is only a set of values and the only relation in this domain is the equality relation. The domain of a numerical attribute are the real numbers and this domain is much more informative: it has the structure of linear order and the natural metric, i.e., the absolute difference.

Below we define an important property of metrics related to numerical attributes:

Definition 2. *The metric ρ is* consistent with the natural linear order of numerical values *if and only if for each numerical attribute a_i and for each three real values $v_1 \leq v_2 \leq v_3$ the following conditions hold: $\rho_i(v_1, v_2) \leq \rho_i(v_1, v_3)$ and $\rho_i(v_2, v_3) \leq \rho_i(v_1, v_3)$.*

The values of a numerical attribute reflect usually a measure of a certain natural property of analyzed objects, e.g., size, age or measured quantities like temperature. Therefore, the natural linear order of numerical values helps often obtain useful information for measuring similarity between objects and the notion of metric consistency describes the metrics that preserve this linear order.

Fact 3. *The city-block metric is consistent with the natural linear order.*

Proof. The city-block metric depends linearly on the absolute difference as defined in Equation 2 or 3. Since the absolute difference is consistent with the natural linear order, the city-block metric is consistent too. □

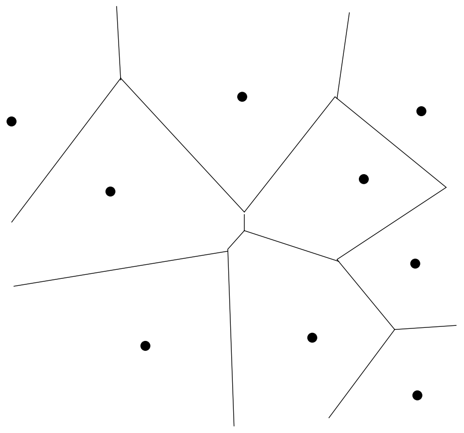


Fig. 1. The Voronoi diagram determined by examples on the Euclidean plane

2.5 *K* Nearest Neighbors as Analogy-Based Reasoning

One of the most popular algorithms in machine learning is the k nearest neighbors (k-nn). The predecessor of this method, the nearest neighbor algorithm (1-nn) [23], induces a metric ρ from the training set U_{trn}, e.g., the city-block and Hamming metric described in Subsection 2.4, and stores the whole training set U_{trn} in memory. Each test object x is classified by the 1-nn with the decision of the nearest training object $y_{nearest}$ from the training set U_{trn} according to the metric ρ:

$$\begin{aligned} y_{nearest} &:= \arg \min_{y \in U_{trn}} \rho(x, y), \\ dec_{1-nn}(x) &:= dec(y_{nearest}). \end{aligned}$$

On the Euclidean plane (i.e., with the Euclidean metric) the regions of the points nearest to particular training examples constitute the Voronoi diagram (see Figure 1).

The k nearest neighbors is an extension of the nearest neighbor [26, 31]. Instead of the one nearest neighbor it uses the k nearest neighbors $NN(x, k)$ to select the decision for an object x to be classified. The object x is assigned with the most frequent decision among the k nearest neighbors:

$$dec_{k-nn}(x) := \arg \max_{d_j \in V_{dec}} \left| \{ y \in NN(x, k) : dec(y) = d_j \} \right|. \tag{4}$$

Ties are broken arbitrary in favor of the decision d_j with the smallest index j or in favor of a randomly selected decision among the ties.

The k nearest neighbors method is a simple example of analogy-based reasoning. In this approach a reasoning system assumes that there is a database providing the complete information about examplary objects. When the system is asked about another object with an incomplete information it retrieves similar (analogous) objects from the database and the missing information is completed on the basis of the information about the retrieved objects.

In the k-nn the induced metric ρ plays the role of a similarity measure. The smaller the distance is between two objects, the more similar they are. It is important for the similarity measure to be defined in such a way that it uses only the information that is available both for the examplary objects in the database and for the object in the query. In the problem of decision learning it means that the metric uses only the values of the non-decision attributes.

2.6 Data Sets

The performance of the algorithms described in this dissertation is evaluated for a number of benchmark data sets. The data sets are obtained from the repository of University of California at Irvine [16]. This repository as the source of benchmark data sets is the most popular in the machine learning community and all the data sets selected to evaluate learning algorithms in this dissertation have been also used by other researchers. This ensures that the presented performance of algorithms can be compared to the performance of other methods from the literature.

To compare the accuracy of the learning models described in this dissertation (Section 3 and Section 5) 10 benchmark data sets were selected (see Table 1). All the selected sets are the data sets from UCI repository that have data objects represented as vectors of attributes values and have the size between a few thousand and several tens thousand of objects. This range of the data size was chosen because such data sets are small enough to perform multiple experiments for all the algorithms described in this dissertation and to measure their accuracy in a statistically significant way (see Subsection 2.7). The evaluation of these algorithms is based on the largest possible data sets since such data sets are usually provided in real-life problems.

To compare the efficiency of the indexing structures used to speedu up searching for the nearest neighbors (Section 4) all the 10 data sets from Table 1 were used again with 2 additional very large data sets (see Table 2). The size of the 2 additional data sets is several hundred thousand. The indexing and the searching

Table 1. The data sets used to evaluate accuracy of learning algorithms

Data set	Number of attributes	Types of attributes	Training set size	Test set size
segment	19	numeric	1 540	770
splice (DNA)	60	nominal	2 000	1 186
chess	36	nominal	2 131	1 065
satimage	36	numeric	4 435	2 000
mushroom	21	numeric	5 416	2 708
pendigits	16	numeric	7 494	3 498
nursery	8	nominal	8 640	4 320
letter	16	numeric	15 000	5 000
census94	13	numeric+nominal	30 160	15 062
shuttle	9	numeric	43 500	14 500

Table 2. The data sets used to evaluate efficiency of indexing structures

Data set	Number of attributes	Types of attributes	Training set size	Test set size
census94-95	40	numeric+nominal	199 523	99 762
covertype	12	numeric+nominal	387 308	193 704

process are less time consuming than some of the learning models. Therefore, larger data sets are possible to be tested. The 2 largest data sets illustrate the capabilities of the indexing methods described in the dissertation.

Each data set is split into a training and a test set. Some of the sets (*splice, satimage, pendigits, letter, census94, shuttle, census94-95*) are available in the repository with the original partition and this partition was used in the experiments. The remaining data sets (*segment, chess, mushroom, nursery, covertype*) was randomly split into a training and a test part with the split ratio 2 to 1. To make the results from different experiments comparable the random partition was done once for each data set and the same partition was used in all the performed experiments.

2.7 Experimental Evaluation of Learning Algorithms

Both in the learning models constructed from examples (Sections 3 and 5) and in the indexing structures (Section 4) described in the dissertation there are elements of non-determinism: some of the steps in these algorithms depend on selection of a random sample from a training set. Therefore the single test is not convincing about the superiority of one algorithm over another: difference between two results may be a randomness effect. Instead of the single test in each experiment a number of tests was performed for each data set and the average results are used to compare algorithms. Moreover, the Student's t-test [41, 30] is applied to measure statistical significance of difference between the average results of different algorithms.

The Student's t-test assumes that the goal is to compare two quantities being continuous random variables with normal distribution. A group of sample values is provided for each quantity to be compared. In the dissertation these quantities are either the accuracy of learning algorithms measured on the test set (see Subsection 2.1) or the efficiency of the indexing and searching algorithm measured by the number of basic operations performed.

There are the paired and the unpaired Student's t-test. The paired t-test is used where there is a meaningful one-to-one correspondence between the values in the first and in the second group of sample values to be compared. In our experiments the results obtained in particular tests are independent. In such a situation the unpaired version of the Student's t-test is appropriate.

Another type of distinction between different tests depends on the information one needs to obtain from a test. The one-tailed t-test is used if one needs to know whether one quantity is greater or less than another one. The two-tailed t-test is used if the direction of the difference is not important, i.e., the infor-

Table 3. The Student's t-test probabilities

df \ α	**90%**	**95%**	**97.5%**	**99%**	**99.5%**
1	3.078	6.314	12.706	31.821	63.657
2	1.886	2.920	4.303	6.965	9.925
3	1.638	2.353	3.182	4.541	5.841
4	1.533	2.132	2.776	3.747	4.604
5	1.476	2.015	2.571	3.365	4.032
6	1.440	1.943	2.447	3.143	3.707
7	1.415	1.895	2.365	2.998	3.499
8	1.397	1.860	2.306	2.896	3.355
9	1.383	1.833	2.262	2.821	3.250
10	1.372	1.812	2.228	2.764	3.169

mation whether two quantities differ or not is required only. In our experiments the information about the direction of difference (i.e., whether one algorithm is better or worse than another one) is crucial so we use the one-tailed unpaired Student's t-test.

Let X_1 and X_2 be continuous random variables and let p be a number of values sampled for each variable X_i. In the Student's t-test only the means $E(X_1), E(X_2)$ and the standard deviations $\sigma(X_1), \sigma(X_2)$ are used to measure statistical significance of difference between the variables. First, the value of t is to be calculated:

$$t = \frac{E(X_1) - E(X_2)}{\sqrt{\frac{\sigma(X_1)^2 + \sigma(X_2)^2}{p}}}.$$

Next, the degree of freedom *df* is to be calculated:

$$df = 2(p-1).$$

Now the level of statistical significance can be checked in the table of the t-test probabilities (see Table 3). The row with the calculated degree of freedom *df* is to be used. If the calculated value of t is greater than the critical value of t given in the table then X_1 is greater than X_2 with the level of significance α given in the header of the column. The level of significance α means that X_1 is greater than X_2 with the probability α.

3 Metrics Induced from Examples

This section explores metrics induced from examples.

3.1 Joint City-Block and Value Difference Metric

Subsection 2.4 provides the metric definition that combines the city-block metric for numerical attributes and the Hamming metric for nominal attributes. In this subsection we focus on nominal attributes.

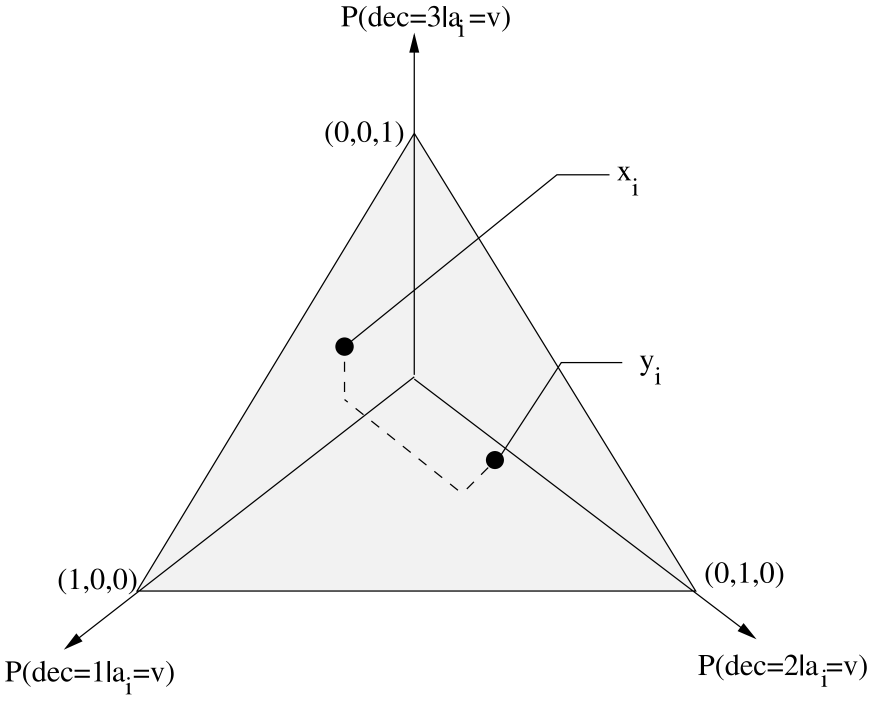


Fig. 2. An example: the Value Difference Metric for the three decision values $V_{dec} = \{1, 2, 3\}$. The distance between two nominal values x_i, y_i corresponds to the length of the dashed line.

The definition of the Hamming metric uses only the relation of equality in the domain of values of a nominal attribute. This is the only relation that can be assumed in general about nominal attributes. This relation carries often insufficient information, in particular it is much less informative than the structure of the domains for numerical attributes where the values have the structure of linear order with a distance measure between the values.

Although in general one can assume nothing more than equality relation on nominal values, in the problem of learning from examples the goal is to induce a classification model from examples assuming that a problem and data are fixed. It means that in the process of classification model induction the information encoded in the database of examples should be used. In the k nearest neighbors method this database can be used to extract meaningful information about relation between values of each nominal attribute and to construct a metric.

This fact has been used first by Stanfill and Waltz who defined a measure to compare the values of a nominal attribute [77]. The definition of this measure, called the Value Difference Metric (VDM), is valid only for the problem of learning from examples. It defines how much the values of a nominal attribute $a_i \in A$ differ in relation to the decision *dec*. More precisely, the VDM metric estimates the conditional decision probability $P(dec = d_j|a_i = v)$ given a nominal value v and uses the estimated decision probabilities to compare nominal values. The VDM distance between two nominal values x_i, y_i is defined by the difference between the estimated decision probabilities $P(dec = d_j|a_i = x_i)$, $P(dec = d_j|a_i = x_i)$ corresponding to the values x_i, y_i (see Figure 2):

$$\rho_i(x_i, y_i) = \sum_{d_j \in V_{dec}} |P(dec = d_j|a_i = x_i) - P(dec = d_j|a_i = y_i)| . \tag{5}$$

The estimation of the decision probability $P(dec = d_j|a_i = v)$ is done from the training set U_{trn}. For each value v, it is defined by the decision distribution in the set of all the training objects that have the value of the nominal attribute a_i equal to v:

$$P_{VDM}(dec = d_j|a_i = v) = \frac{|\{x \in U_{trn} : dec(x) = d_j \wedge x_i = v\}|}{|\{x \in U_{trn} : x_i = v\}|}.$$

From Equation 5 and the definition of $P_{VDM}(dec = d_j|a_i = v)$ one can see that the more similar the correlations between each of two nominal values $x_i, y_i \in V_i$ and the decisions $d_1, \ldots, d_m \in V_{dec}$ in the training set of examples U_{trn} are the smaller the distance in Equation 5 is between x_i and y_i. Different variants of these metric were used in many applications [14, 22, 77].

To define a complete metric the Value Difference Metric needs to be combined with another distance function for numerical attributes. For each pair of possible data objects $x, y \in \mathbb{X}$ the following condition $\rho_i(x_i, y_i) \leq 2$ is satisfied for any nominal attribute $a_i \in A$. It means that the range of possible distances for the values of nominal attributes in the Value Difference Metric is $[0; 2]$. It corresponds well to the city-block distance for a numerical attribute a_i normalized by the range of the values of this attribute in the training set U_{trn} (see Subsection 2.4):

$$\rho_i(x_i, y_i) = \frac{|x_i - y_i|}{max_i - min_i}.$$

The range of this normalized city-block metric is $[0; 1]$ for the objects in the training set U_{trn}. In the test set U_{tst} this range can be exceeded but it happens very rarely in practice. The most important property is that the ranges of such a normalized numerical metric and the VDM metric are of the same order.

The above described combination of the distance functions for nominal and numerical attributes was proposed by Domingos [25]. The experimental results described in Subsection 3.7 and 3.9 prove that this combination is more effective than the same normalized city-block metric combined with the Hamming metric.

3.2 Extensions of Value Difference Metric for Numerical Attributes

The normalized city-block metric used in the previous subsection to define the joint metric uses information from the training set: it normalizes the difference between two numerical values v_1, v_2 by the range of the values of a numerical attribute $max_i - min_i$ in the training set. However, it defines the distance between values of the numerical attribute on the basis of the information about this attribute only, whereas the distance definition for nominal attributes makes use of the correlation between the nominal values of an attribute and the decision values. Since this approach improves the effectiveness of metrics for nominal attributes (see Subsection 3.7) analogous solutions has been investigated for numerical attributes.

Wilson and Martinez proposed two analogous distance definitions. In the Interpolated Value Difference Metric (IVDM) [88, 89] it is assumed that the range of values $[min_i; max_i]$ of a numerical attribute a_i in a training set is discretized into s equal-width intervals. To determine the value of s they use the heuristic value

$$s = \max\left(|V_{dec}|, 5\right).$$

The width of such a discretized interval is:

$$w_i = \frac{max_i - min_i}{s}.$$

In each interval $I_p = [min_i + (p-1) \cdot w_i;\ min_i + p \cdot w_i]$, where $0 \leq p \leq s+1$, the midpoint mid_p and the decision distribution $P(dec = d_j | a_i \in I_p)$ are defined by

$$mid_p = min_i + (p - \frac{1}{2}) \cdot w_i,$$

$$P(dec = d_j | a_i \in I_p) = \begin{cases} 0 & \text{if } p = 0 \text{ or } p = s+1 \\ \frac{|\{x \in U_{trn} :\, dec(x) = d_j \wedge x_i \in I_p\}|}{|\{x \in U_{trn} :\, x_i \in I_p\}|} & \text{if } 1 \leq p \leq s. \end{cases}$$

.

To determine the decision distribution in the IVDM metric for a given numerical value v the two neighboring intervals are defined by

$$\underline{I(v)} = \max\{p \leq s+1 :\ mid_p \leq v \vee p = 0\},$$

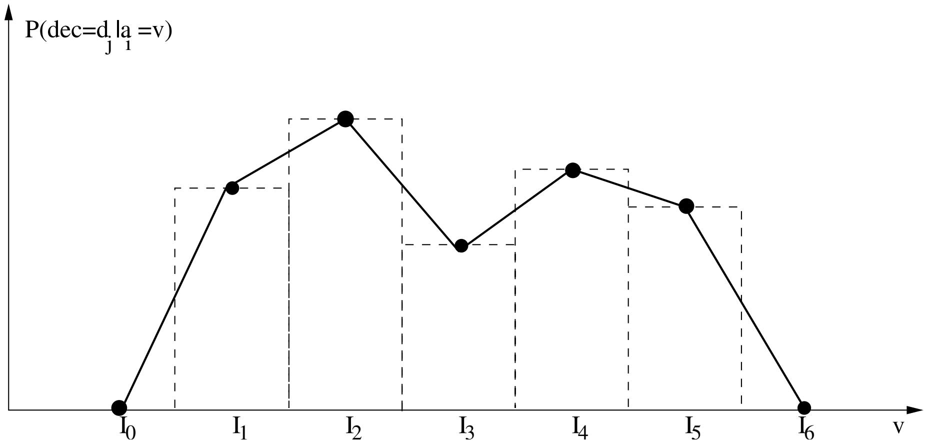


Fig. 3. An example of the interpolated decision distribution for a single decision d_j with the number of intervals $s = 5$

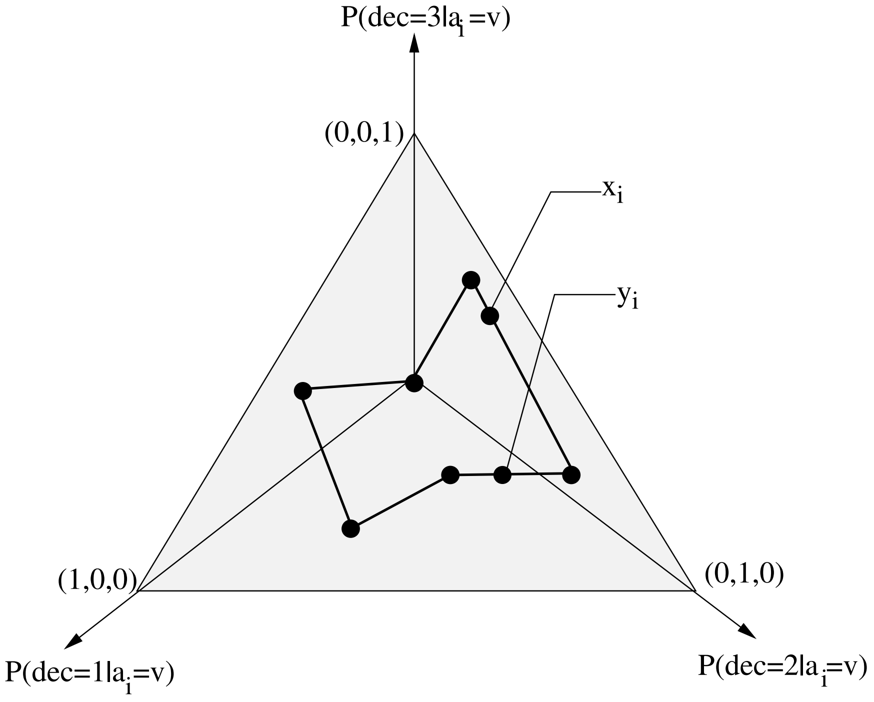


Fig. 4. The Interpolated Value Difference Metric: to measure the distance between two numerical values x_i, y_i the decision distribution for each value is interpolated between the decision distributions in the midpoints of the two neighboring intervals.

$$\overline{I(v)} = \min\{p \geq 0 : \, mid_p \geq v \, \vee \, p = s+1\}.$$

If v is out of the range $[mid_0; mid_{s+1}]$ the interval indices are set either to zero: $\underline{I(v)} = \overline{I(v)} = 0$ or to $s+1$: $\underline{I(v)} = \overline{I(v)} = s+1$, and the null distribution is assigned to v. If v lies in the range $[mid_0; mid_{s+1}]$ there are two cases. If $\underline{I(v)}$ and $\overline{I(v)}$ are equal the value v is exactly the midpoint of the interval $I_{\underline{I(v)}} = I_{\overline{I(v)}}$ and the decision distribution from this interval $P(dec = d_j | a_i \in I_{\underline{I(v)}})$ is used to compare v with other numerical values. Otherwise, the decision distribution for the value v is interpolated between the two neighboring intervals $I_{\underline{I(v)}}$ and $I_{\overline{I(v)}}$. The weights of the interpolation are proportional to the distances to the midpoints of the neighboring intervals (see Figure 3):

$$P_{IVDM}(dec = d_j | a_i = v) =$$
$$P(dec = d_j | a_i \in I_{\underline{I(v)}}) \cdot \frac{mid_{\overline{I(v)}} - v}{w_i} + P(dec = d_j | a_i \in I_{\overline{I(v)}}) \cdot \frac{v - mid_{\underline{I(v)}}}{w_i}.$$

The decision distributions for the values of a numerical attribute correspond to the broken line in the space of decision distributions in Figure 4. The dimension of this space is equal to the number of decisions $m = |V_{dec}|$. To define the IVDM metric these decision distributions for numerical values are used by analogy to the decision distributions for nominal values of nominal attributes the VDM metric. The IVDM distance between two numerical values is defined by Equation 5 as equal to the city-block distance between the two corresponding distributions in the space of decision distributions.

The IVDM metric can be explained by means of sampling the value of $P(dec = d_j | a_i \in I_p)$ at the midpoint mid_p of each discretized interval $[mid_p - \frac{w_i}{2}; mid_p + \frac{w_i}{2}]$. Then the IVDM metric interpolates between these sampled points to provide a continuous approximation of the decision probability $P(dec = d_j | a_i = v)$ for the whole range of values of the attribute a_i.

The IVDM metric is computationally effective. The limits of the range of values min_i, max_i, the interval width w_i and the decision distributions in the discretized intervals $I_0, \ldots, I_{s+1}$ for all attributes can be computed in linear time $O(|U_{trn}|\,|A|)$. The cost of the single distance computation is also linear $O(|A|\,|V_{dec}|)$: the two neighboring intervals of a value v can be determined in a constant time by the evaluation of the expressions:

$$\underline{I(v)} = \begin{cases} 0 & \text{if } v < min_i - \frac{w_i}{2} \\ s+1 & \text{if } v > max_i + \frac{w_i}{2} \\ \left\lfloor \frac{v - min_i + \frac{w_i}{2}}{w_i} \right\rfloor & \text{if } v \in [min_i - \frac{w_i}{2}; max_i + \frac{w_i}{2}], \end{cases}$$

$$\overline{I(v)} = \begin{cases} 0 & \text{if } v < min_i - \frac{w_i}{2} \\ s+1 & \text{if } v > max_i + \frac{w_i}{2} \\ \left\lceil \frac{v - min_i + \frac{w_i}{2}}{w_i} \right\rceil & \text{if } v \in [min_i - \frac{w_i}{2}; max_i + \frac{w_i}{2}]. \end{cases}$$

and the interpolation of two decision distributions can be computed in $O(|V_{dec}|)$.

Another extension of the VDM metric proposed by Wilson and Martinez is the Windowed Value Difference Metric (WVDM) [88]. It replaces the linear interpolation from the IVDM metric by sampling for each numerical value. The interval width w_i is used only to define the size of the window around the value to be sampled. For a given value v the conditional decision probability $P(dec = d_j | a_i = v)$ is estimated by sampling in the interval $[v - \frac{w_i}{2}; v + \frac{w_i}{2}]$ that v is the midpoint in:

$$P_{WVDM}(dec = d_j | a_i = v) =$$
$$\begin{cases} 0 & \text{if } v \le min_i - \frac{w_i}{2} \text{ or } v \ge max_i + \frac{w_i}{2} \\ \frac{\left|\left\{x \in U_{trn} :\, dec(x) = d_j \wedge |x_i - v| \le \frac{w_i}{2}\right\}\right|}{\left|\left\{x \in U_{trn} :\, |x_i - v| \le \frac{w_i}{2}\right\}\right|} & \text{if } v \in [min_i - \frac{w_i}{2}; max_i + \frac{w_i}{2}]. \end{cases}$$

The WVDM metric locates each value v to be estimated in the midpoint of the interval to be sampled and in this way it provides a closer approximation of the conditional decision probability $P(dec = d_j | a_i = v)$ than the IVDM metric. However, the size of the window is constant. In many problems the density of numerical values is not constant and the relation of being similar between two numerical values depends on the range where these two numerical values occur. It means that the same difference between two numerical values has different meaning in different ranges of the attribute values. For example, the meaning of the temperature difference of the one Celsius degree for the concept of water freezing is different for the temperatures over 20 degrees and for the temperatures close to zero.

Moreover, in some ranges of the values of a numerical attribute the sample from the training set can be sparse and the set of the training objects falling into a window of the width w_i may be insufficiently representative to estimate correctly the decision probability. In the extreme case the sample window can even contain no training objects.

To avoid this problem we propose the Density Based Value Difference Metric (DBVDM) that is a modification of the WVDM metric. In the DBVDM metric the size of the window to be sampled depends on the density of the attribute values in the training set. The constant parameter of the window is the number of the values from the training set falling into the window rather than its width. To estimate the conditional decision probability $P(dec = d_j|a_i = v)$ for a given value v of a numerical attribute a_i the DBVDM metric uses the vicinity set of the value v that contains a fixed number n of objects with the nearest values of the attribute a_i. Let $w_i(v)$ be such a value that

$$\left|\left\{x \in U_{trn} : |v - x_i| < \frac{w_i(v)}{2}\right\}\right| \leq n \text{ and } \left|\left\{x \in U_{trn} : |v - x_i| \leq \frac{w_i(v)}{2}\right\}\right| \geq n.$$

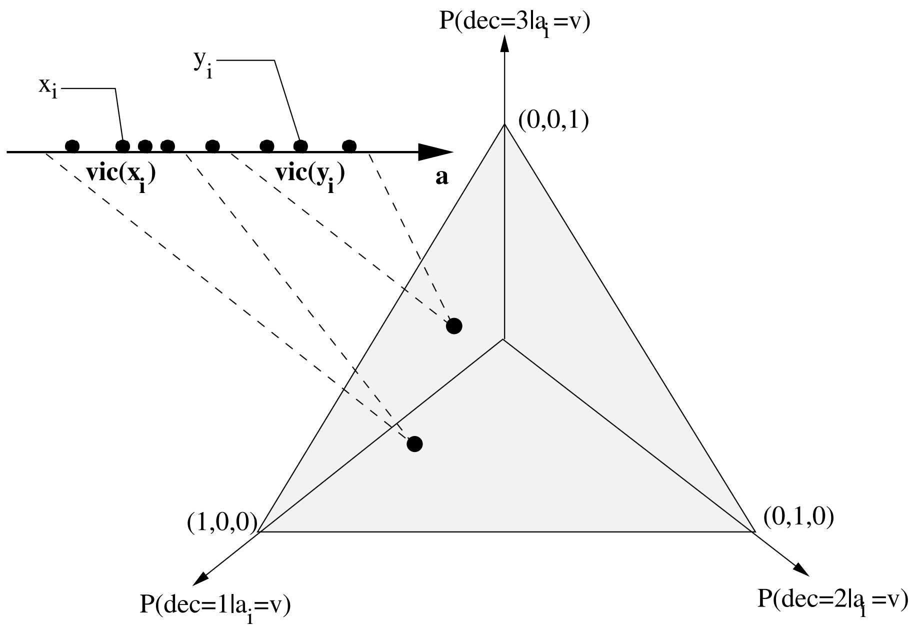


Fig. 5. The Density Based Value Difference Metric: The decision distributions for x_i, y_i are sampled from the windows $vic(x_i), vic(y_i)$ around x_i and y_i, repsectively, with a constant number of values in a training set

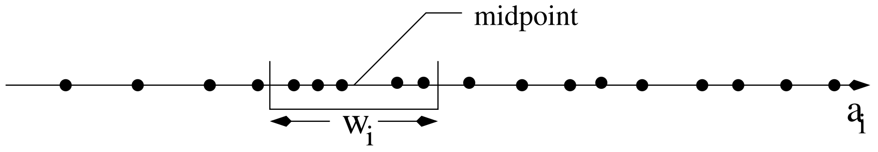


Fig. 6. A window with the midpoint ascending in the domain of values of a numerical attribute a_i

The value $w_i(v)$ is equal to the size of the window around v dependent on the value v. The decision probability in the DBVDM metric for the value v is defined as in the WVDM metric. However, it uses this flexible window size $w_i(v)$ (see Figure 5):

$$P_{DBVDM}(dec = d_j | a_i = v) = \frac{\left|\left\{x \in U_{trn} : dec(x) = d_j \wedge |x_i - v| \leq \frac{w_i(v)}{2}\right\}\right|}{\left|\left\{x \in U_{trn} : |x_i - v| \leq \frac{w_i(v)}{2}\right\}\right|}.$$

The DBVDM metric uses the sample size n as the invariable parameter of the procedure estimating the decision probability at each point v. If the value n is selected reasonably the estimation of the decision probability avoids the problem of having either too few or too many examples in the sample. We performed a number of preliminary experiments and we observed that the value $n = 200$ was large enough to provide representative samples for all data sets and increasing the parameter n above 200 did not improve the classification accuracy.

The WVDM and the DBVDM metric are much more computationally complex than the IVDM metric. The basic approach where the estimation of the decision probability for two numerical values v_1, v_2 to be compared is performed during distance computation is expensive: it requires to scan the whole windows around v_1 and v_2 at each distance computation. We propose another solution where the decision probabilities for all values of a numerical attribute are estimated from a training set a priori before any distance is computed.

Theorem 4. *For both metrics WVDM and DBVDM the range of values of a numerical attribute can be effectively divided into $2 \cdot |U_{trn}| + 1$ or less intervals in such a way that the estimated decision probability in each interval is constant.*

Proof. Consider a window in the domain of real values moving in such a way that the midpoint of this window is ascending (see Figure 6). In case of the WVDM metric the window has the fixed size w_i. All the windows with the midpoint $v \in \left(-\infty; min_i - \frac{w_i}{2}\right)$ contain no training objects. While the midpoint of the window is ascending in the interval $\left[min_i - \frac{w_i}{2}; max_i + \frac{w_i}{2}\right]$ the contents of the window changes every time when the lower or the upper limit of the window meets a value from the training set U_{trn}. The number of different values in U_{trn} is at most $|U_{trn}|$. Hence, each of the two window limits can meet a new value at most $|U_{trn}|$ times. Hence, the contents of the window can change at most $2 \cdot |U_{trn}|$ times. Since the decision probability estimation is constant if the contents of the window does not change there are at most $2 \cdot |U_{trn}| + 1$ intervals each with constant decision probability.

In the DBVDM metric at the beginning the window contains a fixed number of training objects with the lowest values of the numerical attribute to be considered. Formally, while the midpoint of the window is ascending in the range $\left(-\infty; min_i + \frac{w_i(min_i)}{2}\right)$ the upper limit of the window is constantly equal to $min_i + w_i(min)$ and the lower limit is ascending. Consider the midpoint ascending in the interval

$\left[min_i + \frac{w_i(min_i)}{2}; max_i - \frac{w_i(max_i)}{2}\right]$. In DBVDM the size of the window is changing but one of the two limits of the window is constant. If the lower limit has recently met an object from the training set then it is constant and the upper limit is ascending. If the upper limit meets an object it becomes constant and the lower limit starts to ascend. This repeats until the upper limit of the window crosses the maximum value max_i. Hence, as in WVDM, the contents of the window can change at most $2 \cdot |U_{trn}|$ times and the domain of numerical values can be divided into $2 \cdot |U_{trn}| + 1$ intervals each with constant decision probability. $\square$

Given the list of the objects from the training set sorted in the ascending order of the values of a numerical attribute a_i the proof provides a linear procedure for finding the intervals with constant decision probability. The sorting cost dominates therefore the decision probabilities for all the values of all the attributes can be estimated in $O(|A|\,|U_{trn}| \log |U_{trn}|)$ time. To compute the distance between two objects one needs to find the appropriate interval for each numerical value in these objects. A single interval can be found with the binary search in $O(\log |U_{trn}|)$ time. Hence, the cost of a single distance computation is $O(|A|\,|V_{dec}| \log |U_{trn}|)$. If the same objects are used to compute many distances the intervals corresponding to the attribute values can be found once and the pointers to these intervals can be saved.

All the metrics presented in this subsection: IVDM, WVDM and DBVDM use the information about the correlation between the numerical values and the decision from the training set. However, contrary to the city-block metric none of those three metrics is consistent with the natural linear order of numerical values (see Definition 2). Summing up, the metrics IVDM, WVDM and DBVDM are based more than the city-block metric on the information included in training data and less on the general properties of numerical attributes.

3.3 Weighting Attributes in Metrics

In the previous subsections we used the distance defined by Equation 1 without attribute weighting. This definition treats all attributes as equally important. However, there are numerous factors that make attributes unequally significant for classification in most real-life data sets. For example:

– some attributes can be strongly correlated with the decision while other attributes can be independent of the decision,
– more than one attribute can correspond to the same information, hence, taking one attribute into consideration can make other attributes redundant,

– some attributes can contain noise in values, which makes them less trustworthy than attributes with the exact information.

Therefore, in many applications attribute weighting has a significant impact on the classification accuracy of the k-nn method [2, 51, 56, 85]. To improve the quality of the metrics described in Subsection 3.2 we also use attribute weighting and we replace the non-weighted distance definition from Equation 1 with the weighted version:

$$\rho(x, y) = \sum_{i=1}^{n} w_i \cdot \rho_i(x_i, y_i). \tag{6}$$

In the dissertation we combine attribute weighting with linear metrics. As we substantiated in Subsection 2.3 the linear metric is the optimal trade-off between the quality of the measure and its properties.

Weighting methods can be categorized along several dimensions [85]. The main criterion for distinction depends on whether a weighting algorithm computes the weights once following a pre-existing model or uses feedback from performance of a metric to improve weights iteratively. The latter approach has an advantage over the former one: the search for weight settings is guided by estimation how well those settings perform. Thus, attribute weights are adjusted to data more than in case of a fixed, pre-existing model. In this dissertation we propose the weighting methods that incorporate performance feedback.

The next distinction among algorithms searching in a weight space depends on the form of a single step in an algorithm. The algorithms fall into two categories:

– on-line algorithms: training examples are processed sequentially and the weights are modified after each example; usually the weights are modified in such a way that the distance to nearby examples from the same class is decreased and the distance to nearby examples from other classes is increased,
– batch algorithms: the weights are modified after processing either the whole training set or a selected sample from the training set.

Online algorithms change weights much more often than batch algorithms so they require much less examples to process. However, for large data sets both online and batch algorithms are too expensive and an advanced indexing method must be applied (see Section 4). In such a case online algorithms are impractical: indexing must be performed every time when weights are modified, in online algorithms it is after each example. Therefore we focus our research on batch methods. Batch algorithms have the additional advantage: online algorithms are sensitive to an order of training examples, whereas batch algorithms are not.

Lowe [56] and Wettschereck [84] have proposed such batch algorithms using performance feedback. Both algorithms use the conjugate gradient to optimize attribute weights in order to minimize a certain error function based on the leave-one-out test on a training set. However, Lowe and Wettischereck's methods are applicable only to the specific weighted Euclidean metric. To make it possible to apply attribute weighting to different metrics we propose and test two batch

Algorithm 1. Attribute weighting algorithm optimizing distance

```
$nearest(x)$ - the nearest neighbor of $x$ in the sample $S_{trn}$

for each attribute $w_i := 1.0$
$modifier := 0.9$
$convergence := 0.9$
repeat $l$ times
    $S_{trn} :=$ a random training sample from $U_{trn}$
    $S_{tst} :=$ a random test sample from $U_{trn}$
    $MR := \frac{\sum_{x \in S_{tst} : dec(x) \neq dec(nearest(x))} \rho(x, nearest(x))}{\sum_{x \in S_{tst}} \rho(x, nearest(x))}$
    for each attribute $a_i$
        $MR(a_i) := \frac{\sum_{x \in S_{tst} : dec(x) \neq dec(nearest(x))} \rho_i(x_i, nearest(x)_i)}{\sum_{x \in S_{tst}} \rho_i(x_i, nearest(x)_i)}$
    for each attribute $a_i$
        if $MR(a_i) > MR$ then $w_i := w_i + modifier$
    $modifier := modifier \cdot convergence$
```

methods based on less restrictive assumptions. They assume only that metrics are defined by the linear combination of metrics for particular attributes as in Equation 6. The first proposed method optimizes distance to the objects classifying correctly in a training set and the second one optimizes classification accuracy in a training set.

A general scheme of those algorithms is the following: they start with the initial weights $w_i := 1$, and iteratively improve the weights. At each iteration the algorithms use the distance definition from Equation 6 with the weights w_i from the previous iteration.

3.4 Attribute Weighting Method Optimizing Distance

Algorithm 1 presents the weighting method optimizing distance. At each iteration the algorithm selects a random training and a random test samples S_{trn} and S_{tst}, classifies each test object x from S_{tst} with its nearest neighbor in S_{trn} and computes the global misclassification ratio MR and the misclassification ratio $MR(a_i)$ for each attribute a_i. The misclassification ratio is the ratio between the sums of the distances to the nearest neighbors $\rho(x, nearest(x))$ for the incorrectly classified objects and for all training objects, respectively. Attributes with greater misclassification ratio $MR(a_i)$ than others have a larger share in the distance between incorrectly classified objects and their nearest neighbors. All attributes a_i that have the misclassification ratio $MR(a_i)$ higher than the global misclassification ratio MR have the weights w_i increased.

If the misclassification ratio $MR(a_i)$ of an attribute a_i is large then the distance between incorrectly classified objects and their nearest neighbors is influenced by the attribute a_i more than the distance between correctly classified objects and their nearest neighbors. The goal of weight modification is

Algorithm 2. Attribute weighting algorithm optimizing classification accuracy

```
$nearest(x)$ - the nearest neighbor of $x$ with the same decision
               in the sample $S_{trn}$
$\overline{nearest}(x)$ - the nearest neighbor of $x$ with a different decision
               in the sample $S_{trn}$

for each attribute $w_i := 1.0$
$modifier := 0.9$
$convergence := 0.9$
repeat $l$ times
  $S_{trn} :=$ a random training sample from $U_{trn}$
  $S_{tst} :=$ a random test sample from $U_{trn}$
  $correct := \left|\left\{x : \rho(x, nearest(x)) \leq \rho(x, \overline{nearest}(x))\right\}\right|$
  for each attribute $a_i$
    $correct(a_i) := \left|\left\{x : \rho_i(x_i, nearest(x)_i) \leq \rho_i(x_i, \overline{nearest}(x)_i)\right\}\right|$
  for each attribute $a_i$
    if $correct(a_i) > correct$ then $w_i := w_i + modifier$
  $modifier := modifier \cdot convergence$
```

to replace incorrectly classifying nearest neighbors without affecting correctly classifying nearest neighbors. Increasing the weights of attributes with the large misclassification ratio gives a greater chance to reach this goal than increasing the weights of attributes with the small misclassification ratio.

In order to make the procedure convergable the coefficient *modifier* used to modify the weights is decreased at each iteration of the algorithm. We performed a number of preliminary experiments to determine the appropriate number of iterations l. It is important to balance between the optimality of the final weights and the time of computation. For all tested data sets we observed that increasing the number of iterations l above 20 did not improve the results significantly and on the other hand the time of computations with $l = 20$ is still acceptable for all sets. Therefore in all further experiments we set the number of iterations to $l = 20$.

3.5 Attribute Weighting Method Optimizing Classification Accuracy

Algorithm 2 presents the weighting method optimizing classification accuracy. At each iteration the algorithm selects a random training and a random test samples S_{trn} and S_{tst} and for each test object x from S_{tst} it finds the nearest neighbor $nearest(x)$ with the same decision and the nearest neighbor $\overline{nearest}(x)$ with a different decision in S_{trn}. Then for each attribute a_i it counts two numbers. The first number *correct* is the number of objects that are correctly classified with their nearest neighbors according to the total distance ρ, i.e., the objects for which the nearest object with the correct decision $nearest(x)$ is closer than the nearest object with a wrong decision $\overline{nearest}(x)$. The second number $correct(a_i)$ is the number of objects for which the component $\rho_i(x_i, nearest(x)_i)$ related to the attribute a_i in the distance to the correct nearest neighbor $\rho(x, nearest(x))$ is less than the corresponding component $\rho_i(x_i, \overline{nearest}(x)_i)$ in the distance to the wrong nearest neighbor $\rho(x, \overline{nearest}(x))$. If the number of objects correctly classified by a particular attribute a_i ($correct(a_i)$) is greater than the number of objects correctly classified by the total distance ($correct$), the weight for this attribute w_i is increased. Like in the previous weighting algorithm to make the procedure convergable the coefficient $modifier$ used to modify the weights is decreased at each iteration and the number of iterations is set to $l = 20$ in all experiments.

3.6 Experiments

In the next subsections we compare the performance of the k nearest neighbors method for the metrics and the weighting methods described in the previous subsections. We compare the Hamming metric (Subsection 2.4) and the Value Difference Metric (Subsection 3.1) for nominal attributes and the city-block metric (Subsection 2.4), the Interpolated Value Difference Metric and the Density Based Value Difference Metric (Subsection 3.2) for numerical attributes. Comparison between the Interpolated and the Windowed Value Difference Metric (Subsection 3.2) was presented in [88] and the authors reported that there was no significant difference between both metrics. Since the interpolated version is more efficient it was chosen to be compared in this dissertation. As the attribute weighting models we compare the algorithm optimizing distance, the algorithm optimizing classification accuracy and the model without weighting, i.e., all weights are equal $w_i := 1$.

To compare the metrics and the weighting methods we performed a number of experiments for the 10 benchmark data sets presented in Table 1. Each data set was partitioned into a training and a test set as described in Subsection 2.7 and the test set was classified by the training set with the k nearest neighbors method. Each data set was tested 5 times with the same partition and the average classification error is used for comparison. To compare accuracy we present the classification error for $k = 1$ and for k with the best accuracy for each data set. The results for k with the best accuracy are computed in the following way. In each test the classification error was computed for each value of k in the range $1 \leq k \leq 200$ and the smallest error among all k was chosen to compute the average error from 5 tests. It means that for the same data set the results of particular tests can correspond to different values of k.

3.7 Results for Data with Nominal Attributes Only

First, we compare the metrics and the weighting methods for data only with nominal attributes. Among the 10 described data sets there are 3 sets that contain only nominal attributes: *chess*, *nursery* and *splice*.

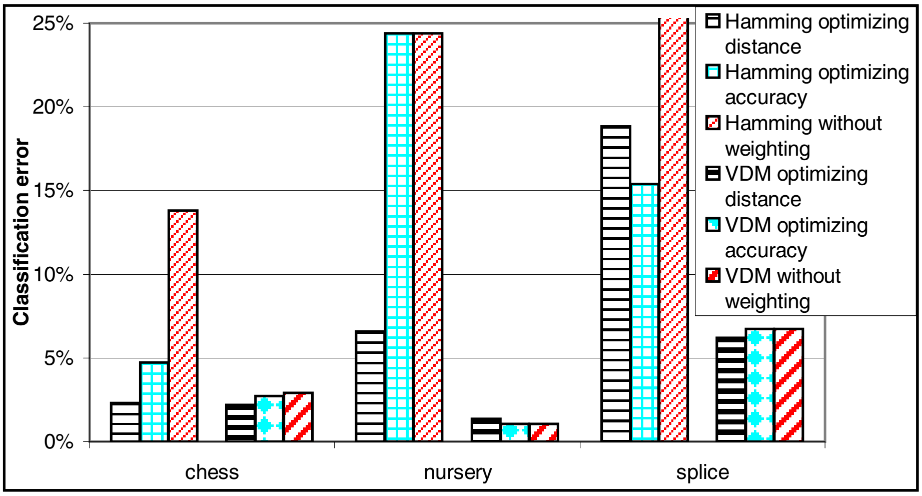


Fig. 7. The average classification error of the 1-nn for the two metrics: Hamming metric and VDM and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting

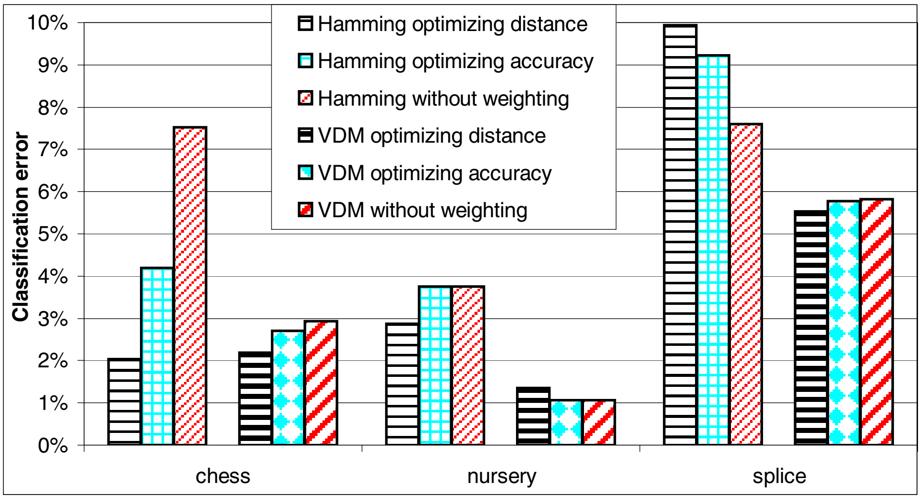


Fig. 8. The average classification error of the k-nn with the best k for the two metrics: Hamming metric and VDM and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting

Figure 7 presents the average classification error of the nearest neighbor method for those 3 data sets. The graph presents the results for the two metrics and for the three weighting methods.

The results for the VDM metric are distinctly better than for the Hamming metric therefore we focus our attention on comparison of the weighting methods for the VDM metric.

The differences between the weighting methods are much smaller but the Student's t-test (see Subsection 2.7) indicates that they are still significant. In case of the data set *chess* the method optimizing distance outperforms the two others the maximum confidence level 99.5%. In case of the data set *nursery* the weighting does not help: the algorithm optimizing classification accuracy gives exactly the same result as without weighting and the algorithm optimizing distance gives a worse result with the confidence level 99.5%. In case of the data set *splice* the algorithm optimizing distance has again the lowest classification error but the statistical significance of the difference is only 97.5%.

Figure 8 presents the average classification error for the best value of k. As in case of the 1-nn, the results for the VDM metric are much better than for the Hamming metric so we compare the weighting methods for the VDM metric.

The table below presents the average value of k with the smallest classification error for particular data sets.

Metric	Hamming			VDM		
Weighting	optimizing distance	optimizing accuracy	none	optimizing distance	optimizing accuracy	none
chess	2.6	1.8	3	1	1	1
nursery	11.6	13	13	1	1	1
splice	152.4	78.4	158	8.2	7	7

In case of the data sets *chess* and *nursery* the average value of the best k for all weighting models for the metric VDM is 1 what means that in all tests the smallest error was obtained for $k = 1$. Hence, for those two data sets and for all the weighting models the average classification error for the best k is equal to the average classification error for $k = 1$ presented before. For *chess* the weighting method optimizing distance outperformed the others with the maximum confidence level 99.5% and for *nursery* the model without weighting provided exactly the same results like the weighting optimizing classification accuracy and both models outperformed the weighting optimizing distance also with the maximum confidence level 99.5%. In case of the data set *splice* the average values of the best k for all the weighting models are greater than 1 so the results are slightly different. In case of 1-nn the weighting optimizing distance is the best but only with the confidence level 97.5% whereas in case of the best k the weighting optimizing distance is the best with the maximum confidence level 99.5%.

The results for data sets with nominal attributes show clearly that the VDM metric is more accurate than the Hamming metric. In case of the Hamming metric the properties of the domain of values of a nominal attribute are only used (the equality relation), whereas in the case of the VDM metric the information contained in a training set is also used. In the latter case a structure of a metric is learnt from a set of values of an attribute in a training set. In comparison to the equality relation such a structure is much richer and it allows to adapt the VDM metric more accurately to data than in the case of the Hamming metric.

The comparison between the weighting methods is not unilateral. However, in most cases the method optimizing distance works best and in case when it loses (for the data set *nursery*) the difference is not so large: the error 1.38% of the weighting optimizing distance in comparison to the error 1.07% of the remaining methods.

3.8 Results for Data with Numerical Attributes Only

In this subsection we present the performance analysis of the metric and weighting models for data only with numerical attributes. There are 6 data sets that contain only numerical attributes: *letter, mushroom, pendigits, satimage, segment* and *shuttle*. All the tests for the data set *mushroom* gave the error 0% and all the tests for the data set *shuttle* gave an error not greater than 0.1%. These two data sets are very easy and the classification results for them can not be a reliable basis for comparison of different metrics and weighting methods. Therefore we exclude those two sets from analysis and we focus on the 4 remaining data sets: *letter, pendigits, satimage* and *segment*.

Figure 9 presents the average classification error of the nearest neighbor method for those 4 data sets. The graph presents the results for the three metrics and for the three weighting methods. First we compare again the metrics. The results are not so unilateral as in case of data with nominal attributes. The table below presents statistical significance of the differences in accuracy between the tested metrics.

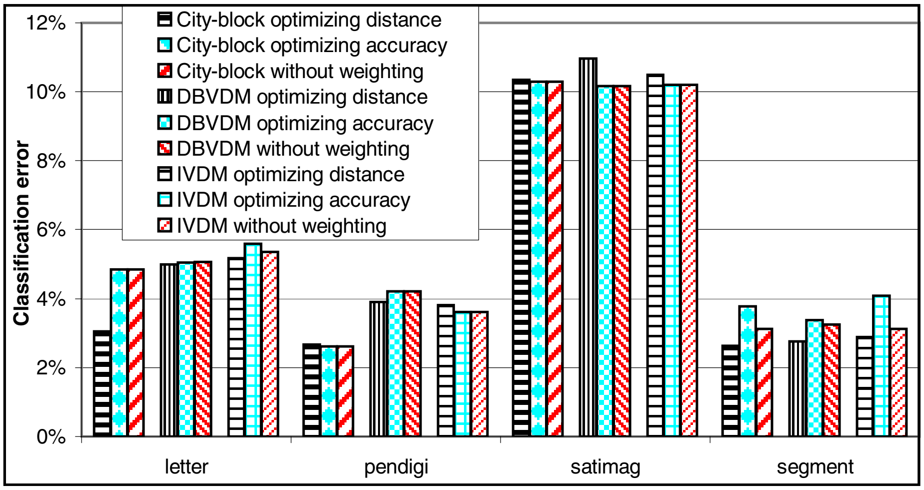


Fig. 9. The average classification error of the 1-nn for the three metrics: The city-block metric, DBVDM and IVDM and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting

Weighting	optimizing distance	optimizing accuracy	none
letter	City-block 99.5%	City-block 99.5%	City-block 99.5%
pendigits	City-block 99.5%	City-block 99.5%	City-block 99.5%
satimage	City-block 90%	DBVDM 99,5%	DBVDM 99,5%
segment	City-block ¡90% (from DBVDM) 90% (from IVDM)	DBVDM 99.5%	City-block & IVDM 99.5%

Each cell in the table presents the metric (or metrics) that the best classification accuracy was obtained for, and explains the confidence level of the difference between this best metric and the other tested metrics for the data set given in the row header and with the weighting method given in the column header. For example, the cell on the crossing of the first row and the first column states that for the data set *letter* with the weighting method optimizing distance the best accuracy was obtained by the city-block metric and the probability that the city-block metric outperforms the others is 99.5%.

The results from the table indicate that the city-block metric wins in most cases, especially when combined with the weighting method optimizing distance. In this case the city-block metric is never worse: for *letter* and *pendigits* it wins with the maximum confidence level 99.5% and for *satimage* and *segment* the classification accuracy for all metrics is similar. In combination with the two other weighting methods the results are not unilateral but still the city-block metric dominates.

If we consider the value of k with the smallest classification error, in each test for the tree data sets: *letter*, *pendigits* and *satimage* it is usually greater than 1. The table below presents the average value of the best k for particular data sets:

Metric	City-block			DBVDM			IVDM		
Weighting	opt. dist.	opt. acc.	none	opt. dist.	opt. acc.	none	opt. dist.	opt. acc.	none
letter	1	5	5	3	2.6	3	1.4	1	1
pendigits	4.6	4	4	3.2	4	4	3.8	4	4
satimage	4.2	3	3	4.6	3	3	3.8	3	3
segment	1.6	1	1	1	1	1	1.4	1	1

Since in tests the best value of k was often greater than 1 the results are different from the case of $k = 1$. Figure 10 presents the average classification error for the best value of k and in the table below we present the winning metric (or metrics) and the confidence level of the difference between the winning metric and the others for the results at Figure 10:

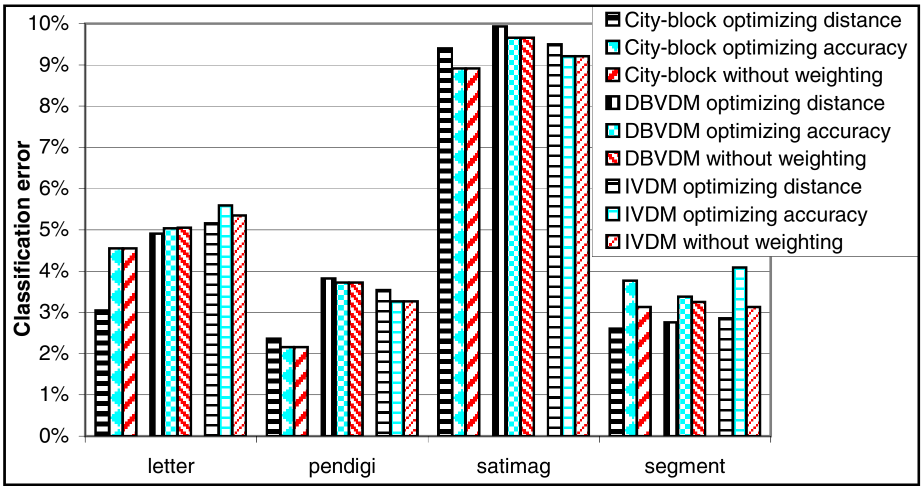


Fig. 10. The average classification error of the k-nn with the best k for the three metrics: The city-block metric, DBVDM and IVDM metric and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting

Weighting	optimizing distance	optimizing accuracy	none
letter	City-block 99.5%	City-block 99.5%	City-block 99.5%
pendigits	City-block 99.5%	City-block 99.5%	City-block 99.5%
satimage	City-block ¡90% (from IVDM) 99.5% (from DBVDM)	City-block 99.5%	City-block 99.5%
segment	City-block ¡90% (from DBVDM) 90% (from IVDM)	DBVDM 99.5%	City-block & IVDM 99.5%

The results are even more unilateral than for the case of $k = 1$. The city-block metric loses only in one case: for the data set *segment* when combined with the weighting method optimizing classification accuracy.

The general conclusion is that the city-block metric is the best for data with numerical attributes and up to now different attempts to replace it with metrics induced from data like the VDM metric for nominal attributes are unsuccessful. This conclusion for numerical data is opposite to the analogous conclusion for nominal data. Like the Hamming metric the city-block metric uses mainly the properties of the domain of values of a numerical attribute. The probable reason for the opposite observation is that the properties of a numerical attribute are much more informative than the equality relation in case of a nominal attribute. In many cases the natural linear order in the domains of numerical attributes corresponds well with the properties of objects important for a decision attribute and the information provided in this linear order is rich enough to work well in the city-block metric. Therefore, it is difficult to construct a better metric from training data. The proposed metrics: DBVDM and IVDM are not consistent with the natural linear order of numerical values. The presented results show that this order is important for reasoning from numerical attributes.

Now, we compare accuracy of the weighting models. The results are presented in Figures 9 and 10 by means of the graphs used for comparison of metrics. The results are different for the different data sets and metrics. However, the city-block metric appeared to be generally the best so we focus on this metric. The table below presents the winning weighting method (or methods) and the confidence level of the difference between this winning method and the others in case of the 1-nn classification and in case of the classification with the best k (using the city-block metric):

k	$k = 1$	the best k
letter	optimizing distance 99.5%	optimizing distance 99.5%
pendigits	optimizing acc. & none ¡90%	optimizing acc. & none 99.5%
satimage	optimizing acc. & none ¡90%	optimizing acc. & none 99.5%
segment	optimizing distance 99% (from none) 99.5% (from optimizing acc.)	optimizing distance 99.5%

The results are not unilateral but we show that the method optimizing distance dominates for the city-block metric. For this metric the results of the method optimizing accuracy differs from the results without weighting only in case of the data set *segment*: the model without weighting provides a better classification. Then it is enough to compare the method optimizing distance to the model without weighting. For $k = 1$, in the cases where the method optimizing distance wins, the statistical significance of the difference is quite large: at least 99%, and the error reduction is also large: from 4.85% to 3.05% for *letter* (37% of the relative difference) and from 3.13% to 2.63% for *segment* (16% of the relative difference) whereas in cases when the method optimizing distance loses the difference is statistically insignificant and relatively very small: 2% of the relative difference for *pendigits* and 0.5% for *satimage*. For the best k all the differences are statistically significant with the maximum confidence level 99.5% but the relative differences are still in favour of the method optimizing distance: for *letter* and *segment* the reduction in error is similar to the case of $k = 1$ (33% and 17% respectively) and for *pendigits* and *satimage* the opposite relative differences in error are only 8% and 5% respectively.

The conclusion is that for the city-block metric it pays to apply the method optimizing distance because a gain in case of improvement can be much larger than a loss in case of worsening. For the two other metrics the results of the weighting methods are more similar and the method optimizing distance does not have the same advantage as in case of the city-block metric.

3.9 Results for Data with Numerical and Nominal Attributes

In this subsection we present analysis of the performance of the metrics and the weighting models for data with both nominal and numerical attributes. There is only one such a data set: *census94*. It is the most difficult data set among all the tested sets: the classification accuracy obtained for *census94* by different classification algorithms from the literature is the lowest [40, 53].

Figure 11 presents the average classification error of the 1-nn for the data set *census94* for all the combinations of the four joint metrics: the Hamming with the city-block, the VDM with the city-block, the VDM with the DBVDM and the VDM with the IVDM metric and the three weighting models: optimizing distance, optimizing accuracy and without weighting. The results are surprising: the best combination is the VDM metric for nominal attributes with the DB-VDM metric for numerical attributes. The same is in the analogous classification results of the k-nn with the best k presented at Figure 12.

Generally, the combinations of the VDM metric with its extensions for numerical attributes: DBVDM and IVDM work better than with the city-block metric. In a sense it is contradictory to the results for data only with numerical attributes. The possible explanation is that the data *census94* are more difficult and the information contained in the properties of the domain of numerical attributes does not correspond directly to the decision. The metrics DBVDM and IVDM are more flexible, they can learn from a training set more than the city-block metric and in case of such difficult data they can adapt more accurately to data.

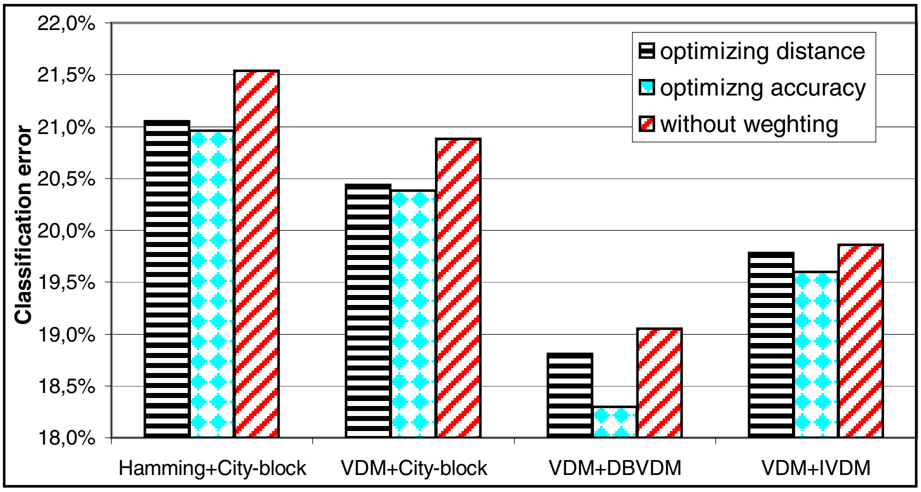


Fig. 11. The average classification error of the 1-nn for the four joint metrics: Hamming with the city-block metric, VDM with the city-block metric, VDM with DBVDM and VDM with IVDM and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting, obtained for the data set *census94*

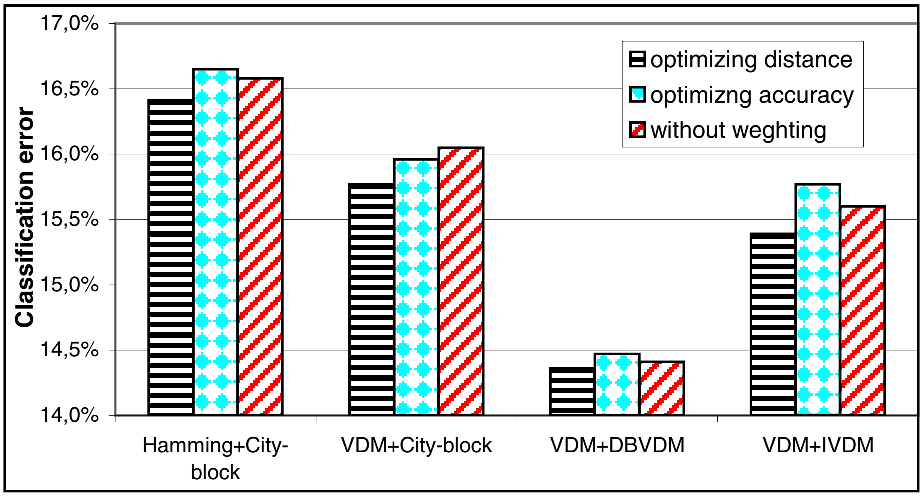


Fig. 12. The average classification error of the k-nn with the best k for the four joint metrics: Hamming with the city-block metric, VDM with the city-block metric, VDM with DBVDM and VDM with IVDM and for the three weighting models: Optimizing distance, optimizing classification accuracy and without weighting, obtained for the data set *census94*

In case of the simpler data sets from Subsection 3.8 the experimental results do not indicate clearly that one of the two metrics DBVDM or IVDM dominates. In case of the data set *census94* the difference between the DBVDM and the IVDM metric is more visible in favour of the DBVDM metric: the classification accuracy of VDM joint with DBVDM is always at least 1% better than the accuracy of VDM joint with IVDM.

Now, we compare the weighting models. The results for $k = 1$ and for the best k are quite different. For $k = 1$ the method optimizing classification accuracy gives the best classification in combination with all the metrics, whereas for the best k the method optimizing distance gives the best accuracy also for all metrics. It is related to the fact that the value of k with the best accuracy is always large for the data set *census94*. The table below presents the average of the best k for all the combinations of the metrics and the weighting methods:

	optimizing distance	optimizing accuracy	none
Hamming+City-block	43.4	23	45
VDM+City-block	34.2	23	27
VDM+DBVDM	84.2	61	83
VDM+IVDM	41a	31	41

The difference in classification between the method optimizing distance and the method optimizing accuracy is not large: in all cases it is below 3% of the relative classification error. The advantage of the method optimizing distance is that in all cases it gives a better result than the model without weighting whereas the method optimizing accuracy is sometimes worse.

3.10 Summary

In this section we have presented the following new methods:

- a Density Based Value Difference Metric for numerical attributes (Subsection 3.2): as distinguished from other metrics of this type from the literature [89, 88] the estimation of the decision probability in DBVDM depends on the density of values,
- an effective method for computing the distance between objects for the metrics WVDM [88] and DBVDM (Subsection 3.2),
- the two attribute weighting batch algorithms using performance feedback, applicable to the whole class of linear metrics: the first one optimizes distance in a training set (Subsection 3.4) and the second one optimizes classification accuracy in a training set (Subsection 3.5).

The experimental results presented in Subsections 3.7, 3.8 and 3.9 lead to the following final conclusions. For nominal attributes the general properties of the domain of values are poor and the information contained in training data is much richer and, therefore, it is important for classification accuracy to incorporate the information from training data into a metric. Hence, a good solution for nominal attributes is the Value Difference Metric described in Subsection 3.1. For numerical attributes the situation is different. The natural linear order provided in the properties of numerical attributes is an important, powerful source of information about objects and in most cases the city-block metric consistent with this natural linear order outperforms the metrics that do not regard this order so strictly. However, the results in Subsection 3.9 show that in cases where data are difficult and the relation between numerical attributes and a decision is not immediate the information contained in data can be still important for the accuracy of a metric. In this case the best classification accuracy has been obtained with use of the DBVDM metric.

In summary, the combination of the VDM metric for nominal attributes and the city-block metric for numerical attributes gives generally the best accuracy and we choose this metric to use in further research: on methods accelerating k nearest neighbors search described in Section 4 and on more advanced metric-based classification models described in Section 5. Since the DBVDM metric was the best for the most difficult data set and in a few other cases, in some experiments in Section 5 we use also the combination of the VDM and the DBVDM metric for comparison.

Comparison of the weighting models does not indicate a particular method to be generally better than others but weighting attributes in a metric usually improves classification accuracy so we decided to choose one for further experiments. Since both for nominal and numerical data the weighting algorithm optimizing distance seems to dominate this one is chosen to be always applied in all further experiments.

Some of the presented metrics and weighting algorithms have been included in the system RSES [8, 73]. The system provides different tools to analyze data, in particular the k nearest neighbors classifier. Two of the presented metrics: the joint VDM and city-block metric and the joint VDM and DBVDM metric and all the three presented weighting models are available in this classifier. They are implemented exactly as described in this section and some of the described parameters of the metrics and the weighting algorithms are available to be set by a user.

4 Distance-Based Indexing and Searching for *k* Nearest Neighbors

Distance-based indexing and the problem of searching for k nearest neighbors is investigated in this section.

4.1 Problem of Searching for *k* Nearest Neighbors

In this section we consider the efficiency problem of the k-nn classifier described in Subsection 2.5. For a long time k-nn was not used in real-life applications due to its large computational complexity. However, the development of methods accelerating searching and the technology advance in recent decade made it possible to apply the method to numerous domains like spatial databases, text information retrieval, image, audio and video recognition, DNA and protein sequence matching, planning, and time series matching (e.g., in stock market prognosis and weather forecasting) [3, 80].

The main time-consuming operation in the k-nn classifier is the distance-based searching for k nearest neighbors of a given query object. Distance-based searching is an extension of the exact searching widely used in text and database applications. It is assumed that a distance measure ρ is defined in a space of objects $\mathbb{X}$ and the problem is to find the set $NN(x, k)$ of k objects from a given training set $U_{trn} \subseteq \mathbb{X}$ that are nearest to a given query object x.

We restrict our consideration to application of k-nn for object classification. It requires fast access to data therefore we concentrate on the case when data are kept in the main memory. With growing size of the main memory in data servers this case attracts more and more attention of people working in different application areas.

The basic approach to searching for k nearest neighbors in a training set is to compute the distance from a query object to each data object in the training set and to select the objects with the smallest distances. The computational cost of finding the nearest neighbors from U_{trn} to all queries in a test set U_{tst} is $O(|U_{tst}|\,|U_{trn}|)$. In many applications the size of a database is large (e.g., several hundred thousand objects) and the cost $O(|U_{tst}|\,|U_{trn}|)$ is not acceptable. This problem is an important issue in many applications therefore it is the subject of the great interest among researchers and practitioners and a considerable effort has been made to accelerate searching techniques and a number of indexing

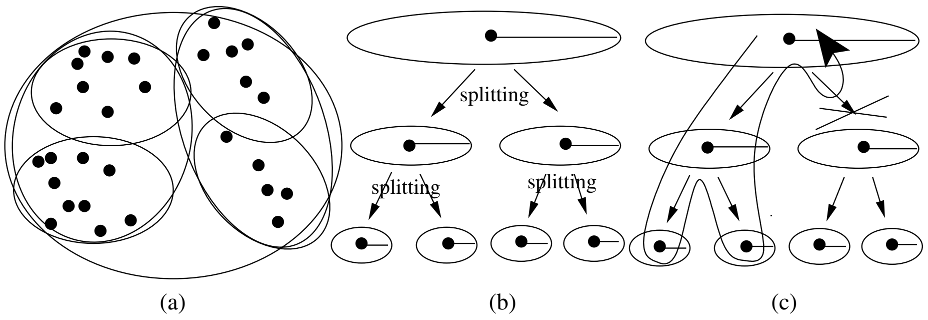


Fig. 13. Indexing and searching in a data set: (a) the hierarchical structure of data clusters (b) the indexing tree with the nodes corresponding to data clusters (c) search pruning in the indexing tree

methods both general and for particular applications have been developed. The most popular idea of indexing is a top-down scheme introduced by Fukunaga and Narendra [35]. It splits the whole training set into clusters in such a way that each cluster contains objects from the same region of a data space (Figure 13a). Each cluster has a compact representation that allows to check quickly whether the cluster can contain the nearest neighbors of a query object (Figure 13b). Instead of comparing a query object directly with each data object first it is compared against the whole regions. If a region is recognized not to contain the nearest neighbors it is discarded from searching. In this way the number of distance computations, and in consequence the performance time, are considerably reduced (Figure 13c).

In the literature one can find indexing methods based on the bottom-up scheme like Ward's clustering [82]. It was recognized that bottom-up constructions lead to a very good performance but instead of reducing the computational cost those bottom-up methods transfer it only from searching to indexing, i.e., searching is much faster but indexing has the $O(|U_{trn}|^2)$ complexity. Hence, this approach is too expensive for most of applications and the top-down scheme has remained the most popular in practice. In the dissertation we focus on the top-down scheme.

An important issue for indexing method construction are the initial assumptions made about a data space. Different models are considered in the literature. The first one assumes that data objects are represented by vectors from a vector space. This model is applicable to databases with numerical attributes or with complex multimedia objects transformable to vectors. It makes it possible to use the algebraic operations on objects in an indexing algorithm: summation and scaling, and construct new objects, e.g., the mean of a set of objects. However, not all databases fit to the model of a vector space. In the dissertation we consider data with both numerical and nominal attributes. The domains of nominal attributes do not have the structural properties used in the model with a vector space. The indexing methods for such data use only a small subset of the properties available in a vector space. Moreover, there are data not based on feature vectors, e.g., texts with the editing distance, DNA or time dependent sequences or plans. The structure of such objects is very specific and for such data the model is limited only by the distance axioms defined in Subsection 2.3. They are sufficient for the k nearest neighbors classification (see Subsection 2.5) therefore many indexing methods in the literature assume the distance model based only on these axioms.

Since a great part of applications is associated with structural databases and multimedia objects transformed to feature vectors a number of indexing techniques have been developed for vector spaces (e.g.,, quad-trees [29] and k-d trees [11]). The cost of a distance computing operation between two vectors is usually low so the methods such as grid-files [62], k-d-b tree [66], R-tree [43] and its variants R^{+}-tree [71] and $R^{\star}$-tree [9] were focused on optimizing the number of I/O operations. The above techniques work well for low dimensional problems, but the performance degrades rapidly with increasing dimensionality. This phenomenon called the dimensional curse have been theoretically substantiated by Beyer et al. [13]. They proved that under certain reasonable assumptions the ratio of the distances to the nearest and the farthest neighbor converges to 1 while increasing the dimension of the vector space. To avoid the problem some specialized methods for high-dimensional spaces have been proposed: X-trees [12], SR-trees [47], TV-trees [55] and VA-files [83].

All the above tree based methods are based on regions in the shape of hypercubes so application of these methods is strictly limited to vector spaces. However, a large number of databases with other kinds of distance measures have raised an increase of interest in general distance-based indexing methods. An exhaustive overview of indexing methods for metric spaces is contained in [19]. SS-tree [86] uses a more general clustering scheme with spheres instead of rectangles as bounding regions but it is still limited to vector spaces because it uses the mean as the center of a cluster. A general distance-based indexing scheme is used in BST [46] and GHT [78]. Both trees have the same construction but different search pruning criteria are used. GNAT [18], SS-tree [86] and M-tree [20] are specialized versions of the BST/GHT tree. To balance the tree GNAT determines separately the number of child nodes for each node. As the splitting procedure GNAT uses the algorithm that selects the previously computed number of centers from a sample and assigns the objects from the parent node to the nearest centers. SS-tree and M-tree are focused on optimizing the number of I/O operations. They maintain a structure of nodes similar to B-trees and assume the dynamic growth of the database. Clustering in M-tree is similar to the clustering algorithm in SS-tree but M-tree uses either a random or a sampled set of the centers instead of the means. Thus, it uses only the distance function and is applicable to any metric space.

All the above mentioned indexing structures from the literature use a one-step clustering procedure to split a node in the tree. Such a procedure selects a number of cluster centers among objects in the given node and assigns each data object from the node to the nearest center. Moreover, the described searching methods use always a single search pruning criterion to accelerate searching in

Algorithm 3. The indexing schema

```
$k$ - the splitting degree of the tree nodes
$root$ - the top node with all the training data objects from $U_{trn}$
$priorityQueue$ - the priority queue of leaf nodes used
                  for the selection of the next node to be split

$priorityQueue := \{root\}$
repeat
   $parent :=$ the next node from $priorityQueue$ to be split
   $splitCluster(parent, k)$
   add $k$ child nodes of $parent$ to $priorityQueue$
until the number of nodes in $priorityQueue$ $\geq \frac{1}{5}|U_{trn}|$
```

an indexing structure. In the next subsections we propose a new method that uses an iterative clustering procedure to split nodes while indexing instead of the one-step procedure and combines three search pruning criteria from BST, GHT and GNAT into one.

We present three versions of this method, depending on the model of data. The first version is appropriate for the model of a vector space, i.e., for data only with numerical attributes. The second variant is appropriate for the model of data considered in the dissertation, i.e., for data with both numerical and nominal attributes. It depends on the metric used to measure distance between data object too. Since the joint city-block and the Value Difference Metric provides the best classification accuracy in the experiments from Section 3 we present the version of indexing that assumes this metric to be used. As the third solution we propose the algorithm based on the most general assumption that only a distance measure is available for indexing.

4.2 Indexing Tree with Center Based Partition of Nodes

Most of the distance based indexing methods reported in the literature [11, 29, 43], [66, 71] and all the methods presented in the paper are based on a tree-like data structure. Algorithm 3 presents the general indexing scheme introduced by Fukunaga and Narendra [35]. All indexing algorithms presented in the paper fit to this scheme. It starts with the whole training data set U_{trn} and splits recursively the data objects into a fixed number k of smaller clusters. The main features that distinguish different indexing trees are the splitting degree of tree nodes k, the splitting procedure *splitCluster* and the pruning criteria used in the search process.

Algorithm 3 assumes that the splitting degree k is the same for all nodes in the tree. An exception to this assumption is Brin's method GNAT [18] that balances the tree by selecting the degree for a node proportional to the number of data objects contained in the node. However, on the ground of experiments Brin concluded that a good balance was not crucial for the performance of the tree. In Subsection 4.4 we present the results that confirm this observation.

Algorithm 4. The iterative k-centers splitting procedure $splitCluster(objects, k)$

```
$objects$ - a collection of data objects to be split
        into $k$ clusters
$Cl_1, \ldots, Cl_k$ - partition of data objects from $objects$
            into a set of clusters
$centers$ - the centers of the clusters $Cl_1, \ldots, Cl_k$
$prevCenters$ - the centers of clusters
            from the last but one iteration
$getCenter(Cl_j)$ - the procedure computing
              the center of the cluster $Cl_j$

repeat
   $centers :=$ select $k$ initial seeds $c_1, \ldots, c_k$ from $objects$
   for each $x \in objects$
      assign $x$ to the cluster $Cl_j$
      with the nearest center $c_j \in centers$
   $prevCenters := centers$
   $centers := \emptyset$
   for each cluster $Cl_j$
      $c_j := getCenter(Cl_j)$
      add $c_j$ to $centers$
until $prevCenters = centers$
```

We have assumed that the algorithm stops when the number of leaf nodes exceeds $\frac{1}{5}$ of the size of the training set $|U_{trn}|$, in other words when the average size of the leaf nodes is 5. It reflects the trade-off between the optimality of a search process and the memory requirements. To make the search process effective the splitting procedure *splitCluster* has the natural property that data objects that are close each to other are assigned to the same child node. Thus, small nodes at the bottom layer of the tree have usually very close objects, and splitting such nodes until singletons are obtained and applying search pruning criteria to such small nodes do not save many distance comparisons. On the other hand, in our implementation the memory usage for the node representation is 2-3 times larger than for the data object representation so the model with the number of leaf nodes equal to $\frac{1}{5}$ of the number of data objects does not increase memory requirements as significantly as the model where nodes are split until the leafs are singletons and the number of all tree nodes is almost twice as the size of the training data set U_{trn}.

Algorithm 4 presents the iterative splitting procedure $splitCluster(objects, k)$ that generalizes the k-means algorithm. Initially, it selects k objects as the centers $c_1, \ldots, c_k$ of clusters. Then it assigns each object x to the cluster with the nearest center and computes the new centers $c_1, \ldots, c_k$. This assignment procedure is iterated until the same set of centers is obtained in two subsequent iterations.

The procedure *getCenter*$(\cdot)$ computes the center of a cluster of objects. Except for this procedure the presented indexing structure preserves the generality: it uses only the notion of distance. The indexing structure is correct for any definition of the procedure *getCenter*$(\cdot)$. However, the efficiency of searching in this structure depends strongly on how the centers of clusters are defined. Therefore we propose different definitions of the centers, depending on the information about the type of a space of objects.

In case of a vector space we propose the means as the centers of clusters:

$$getCenter(Cl) := \frac{\sum_{x \in Cl} x}{|Cl|}$$

In this case Algorithm 4 becomes the well known k-means procedure. Boley and Savaresi have proved the following property of the 2-means algorithm:

Theorem 5. *[70] If a data set is an infinite set of data points uniformly distributed in a 2-dimensional ellipsoid with the semi-axes of the length 1 and a (*$0 < a < 1$*) the* 2*-means iterative procedure with random selection of initial centers has 2 convergence points: one is locally stable and one is locally unstable. The splitting hyperplanes corresponding to the convergence points pass through the center of the ellipsoid and are orthogonal to the main axes of the ellipsoid. In the stable convergence point the splitting hyperplane is orthogonal to the largest axis (see Figure 14).*

This theorem shows that in an infinite theoretical model the 2-means procedure with random selection of initial centers converges in a sense to the optimal

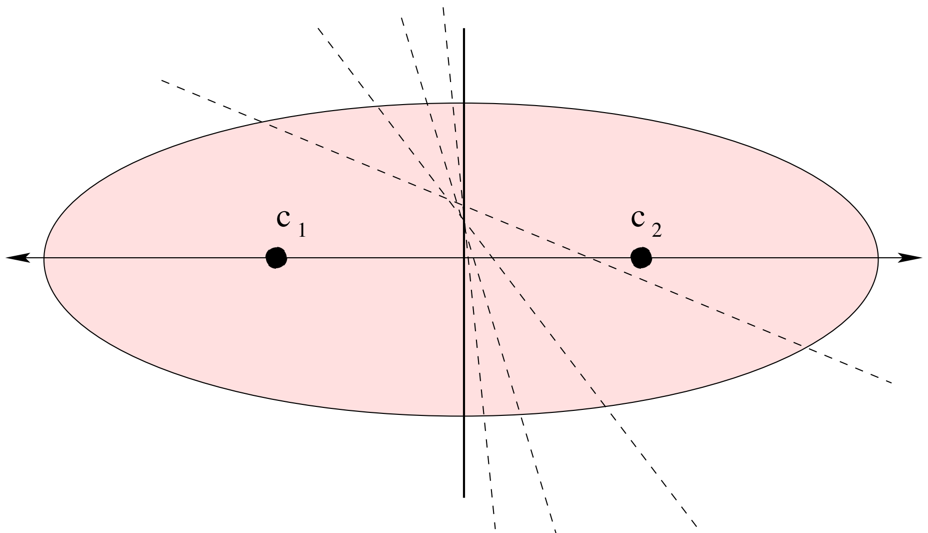


Fig. 14. The convergence of the 2-means procedure to the locally stable partition for data distributed uniformly in an ellipse; the splitting line is orthogonal to the largest axis of the ellipse

partition of data, which may substantiate good splitting properties of this procedure in practice and explain the good experimental performance of the tree based on the 2-means splitting procedure presented in Subsection 4.7.

In the dissertation, we consider data with both numerical and nominal attributes. In Section 3 the joint city-block and VDM metric was proved to provide the best classification accuracy. Therefore, for data with both types of attributes we present the definition of the center of a cluster $getCenter(Cl)$ that assumes this metric to be used for measuring the distance between objects.

The numerical attributes constitute a vector space. Therefore, as in the first version, we propose the mean to be the center value for each numerical attribute a_i:

$$getCenter(Cl)_i := \frac{\sum_{x \in Cl} x_i}{|Cl|}.$$

In case of a nominal attribute the domain of values does not provide the operations of summation and division and the only general property of nominal values is the equality relation. Therefore, as in the problem of metric definition, to define the center of a set of nominal values we use the information encoded in data. Since the centers of clusters are used only by the operation of the distance computation, it is enough to define how the center of nominal values is represented in Equation 5 defining the Value Difference Metric. This equation does not use values directly but it uses the decision probability estimation $P_{VDM}(dec = d_j | a_i = v)$ for each nominal value v. Therefore, for each nominal attribute a_i it is enough to define the analogous decision probability estimation for the center of a set of nominal values:

$$P_{VDM}(dec = d_j | a_i = getCenter(Cl)_i) := \frac{\sum_{x \in Cl} P_{VDM}(dec = d_j | a_i = x_i)}{|Cl|}.$$

This definition of the decision probability estimation for the centers of clusters is correct, because it satisfies the axioms of probability: $P_{VDM}(dec = d_j | getCenter(Cl)_i) \geq 0$ and $\sum_{d_j \in V_{dec}} P_{VDM}(dec = d_j | getCenter(Cl)_i) = 1$. The indexing and searching algorithm use this definition to compute the VDM distance between centers and other objects from a space of objects.

The last version of the procedure $getCenter(\cdot)$ is independent of the metric definition. It is useful in the situation where the model of data does not provide the information how to construct new objects and training objects in U_{trn} are the only objects from a space of objects $\mathbb{X}$ available for an indexing method. In this general case we propose the following approximation of the cluster center. When a cluster Cl contains one or two data objects it selects any of them as the center of Cl. Otherwise the algorithm constructs a sample S that contains the center used to assign objects in the previous iteration of the procedure *splitCluster* and randomly selected $\max(3, \left\lfloor \sqrt{|Cl|} \right\rfloor)$ other objects from Cl. Then it computes the distances among all pairs of objects from S, and selects the object in S that minimizes the second moment of the distance ρ in S, as the new center of Cl:

$$getCenter(Cl) := \arg\min_{x \in S} E\left(\rho(x,y)^2\right).$$

In this way it selects the center from S that minimizes the variance of S. The assumption that the center from the previous iteration is included into the sample S in the next iteration makes it possible to use the previous center in the next center selection. It provides a chance for the stopping condition to be satisfied at each iteration and saves a significant number of unnecessary iterations.

The choice of the value $\max(3, \left\lfloor\sqrt{|Cl|}\right\rfloor)+1$ as the size of the sample S in this center selection algorithm is strictly related to its complexity. A single iteration of the algorithm requires $|S|^2$ distance computations: it computes the distance among all pairs of objects in S. Since the size of the sample S is $O(|Cl|^{\frac{1}{2}})$ the computational cost of a single iteration remains linear with respect to the cluster size $|Cl|$, and thus, it is comparable to the case of the k-means procedure used for vector spaces.

The last algorithm selects the approximate centers for clusters among objects belonging to these clusters. Therefore, in the next subsections we call it the k-approximate-centers algorithm.

The discussion and experimental analysis related to selection of initial centers in Algorithm 4 and the degree of nodes of the presented indexing structure are presented in the next two subsections.

4.3 Selection of Initial Centers

One can consider three general approaches for selection of the initial centers for clusters in the procedure *splitCluster* (see Algorithm 4) known from the literature: random, sampled [18] and exhaustive. The description of BST [46] and GHT [78] is quite general and either it does not specify any particular selection of initial centers or it assumes a simple random model. M- [20] and SS-trees [86] are the dynamic structures and the splitting procedures assume that they operate on an existing inner node of a tree and they have access only to the information contained in a node to be split. While splitting a non-leaf node the algorithm does not have access to all data objects from the subtree of the node so the splitting procedures from M- and SS-trees are incomparable to the presented iterative procedures.

To select the initial centers in GNAT [18] a random sample of the size $3k$ is drown from a set of data objects to be clustered and the initial k centers are picked from this sample. First, the algorithm picks one of the sample data objects at random. Then it picks the sample point that is the farthest from this one. Next, it picks the sample point that is the farthest from these two, i.e., the minimum distance from the two previously picked seeds is the greatest one among all unpicked sample objects. Finally, it picks the point that is the farthest from these three and so on until there are k data points picked.

In the dissertation, we propose yet another method for selecting initial k centers presented in Algorithm 5. It is similar to GNAT's method but it selects the first center more carefully and for selection of the others it uses the whole set to be clustered instead of a sample. Therefore we call this algorithm the

Algorithm 5. The global algorithm for selection of initial centers in the procedure *splitCluster*

```
$c := getCenter(Cl)$
$c_1 := \arg\max_{x \in Cl} \rho(c, x)$
for $j := 2$ to $k$
    $c_j := \arg\max_{x \in Cl} \min_{1 \leq l \leq j-1} \rho(c_l, x)$
```

global selection of the farthest objects. First, the algorithm computes the center c of the whole set to be clustered Cl. As the first seed c_1 it picks the object that is the farthest from the center c of the whole data set. Then it repeats selection of the farthest objects as in GNAT, but from the whole set Cl instead of from a sample. The algorithm can be performed in $O(|Cl|\, k)$ time: it requires to store the minimal distance to selected centers $\min_{1 \leq l \leq j-1} \rho(c_l, x)$ for each object $x \in Cl$ and to update these minimal distances after selection of each next center. For small values of k this cost is acceptable.

One can consider the exhaustive procedure that checks all k-sets among objects to be clustered as the sets of k centers and selects the best one according to a predefined quality measure. However, the computational cost of this method $O(|Cl|^k)$ does not allow us to use it in practice.

Figure 15 presents the performance of the search algorithm for three different seeding procedures used in the k-means based indexing trees with $k = 2$, $k = 3$ and $k = 5$: a simple random procedure, GNAT's sampled selection of the farthest objects and the global selection of the farthest objects described above. The experiments have been performed for the joint city-block and VDM metric with the representation of the center of a cluster extended to nominal attributes as described in the previous subsection, and for the searching algorithm described in Subsections 4.5 and 4.6. All 12 benchmark data sets presented in Tables 1 and 2 have been tested in the following way: the training part of a data set have been indexed with the k-means based indexing tree (once for each combination of $k \in \{2, 3, 5\}$ and the three seeding procedures), and for each object in a test set the two searches have been performed in each indexing tree: for the 1 nearest neighbor and for the 100 nearest neighbors of the test object. At each search the number of distance computations has been counted. The graphs present the average number of distance computations for the whole test set in the 1-nn and the 100-nn search.

The results indicate that the indexing trees with all three methods have comparable performance what may be explained with the good theoretical convergence property of the k-means algorithm formulated in Theorem 5. However, for a few larger data sets: *census94, census94-95, covertype* and *letter* the difference between the global and the two other selection methods is noticeable. In particular, the largest difference is for the data set *census94-95*, e.g., in case of the 2-means based indexing tree the global method takes only 65% of the time of the sampled method and 60% of the time of the random method (as presented at the two upper graphs at Figure 15).

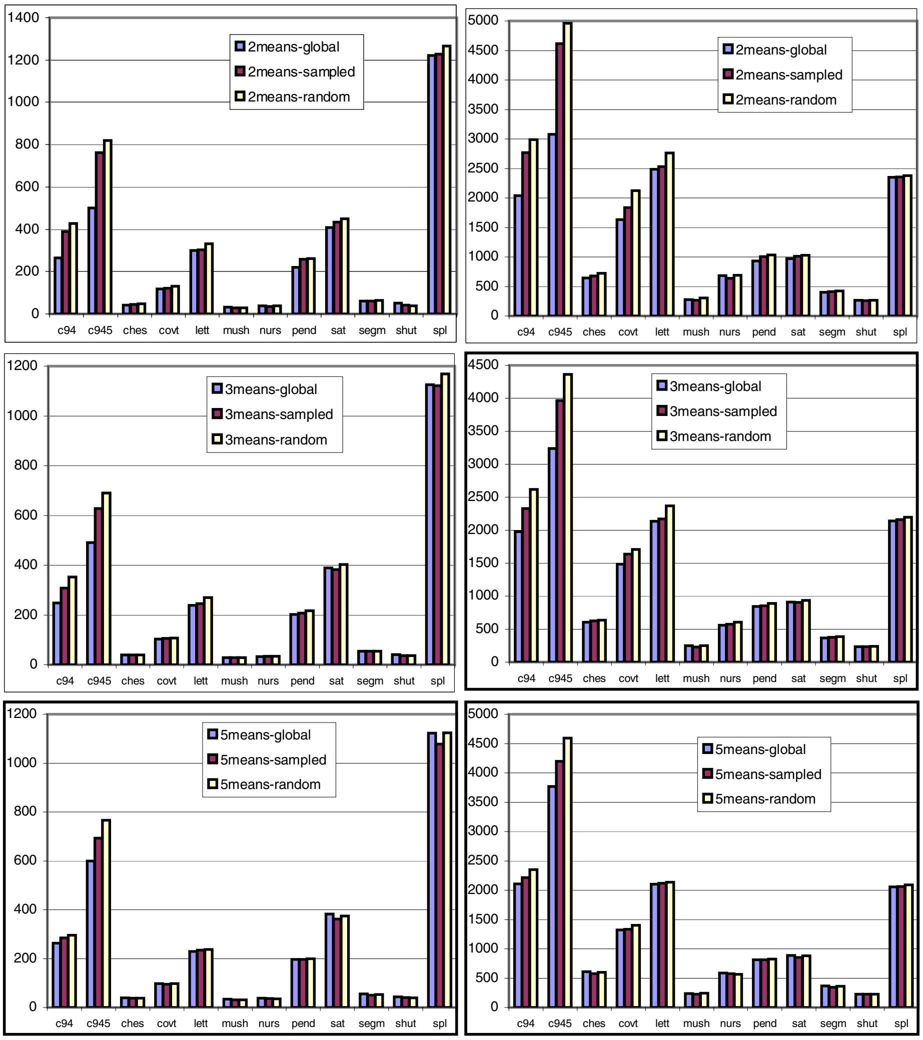


Fig. 15. The average number of distance computations per single object in 1-nn search (the left graphs) and 100-nn search (the right graphs) with the use of 2-means based, 3-means based and 5-means based indexing trees, and with the three different methods of initial center selection: Globally farthest, sampled farthest and random

Summing up, the global method seems to have a little advantage over the others and we decide to use this one in further experiments described in the next subsections.

4.4 Degree of the Indexing Tree

In order to analyze the performance of the k-means based indexing trees, in dependence on the degree of nodes k, we have performed experiments for 8

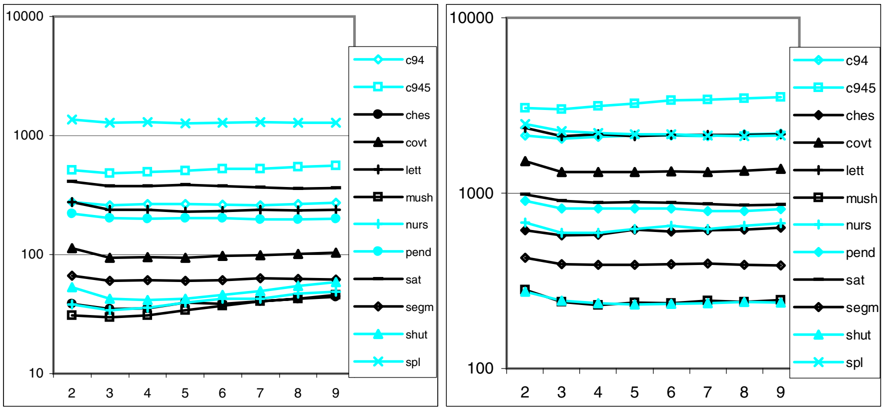


Fig. 16. The average number of distance computations per single object in 1-nn search (the left graph) and 100-nn (the right graph) with the use of the k-means based indexing trees with k in the range $2 \leq k \leq 9$

successive values of k ranging from 2 to 9. Figure 16 presents the performance graphs for particular data sets. As it is shown they are quite stable in the range of tested values except for the value 2 and different values of k have the best performance for particular data sets. For the 1-nn search 7 data sets have the best performance at $k = 3$, 1 at $k = 4$ and 2 at $k = 5$ and $k = 8$. For 100-nn search 4 data sets have the best performance at $k = 3$, 2 at $k = 4, 5$ and 8 and 1 at $k = 7$ and 9. These statistics indicate that the best performance is for small values of k (but greater than 2). Assignment of k to 3, 4 or 5 ensures almost optimal performance.

In the literature the splitting degree of tree nodes is usually assumed to be constant over all nodes in a tree. The exception to this rule is the GNAT structure [18] that attempts to balance the size of branches by choosing different splitting degrees for nodes. It assumes a fixed k to be the average splitting degree of nodes and applies the following procedure to construct a tree. The top node is assigned the degree k. Then each of its child nodes is assigned the degree proportional to the number of data points contained in this child node (with a certain minimum and maximum) so that the average degree of all the child nodes is equal to the global degree k. This process works recursively so that the child nodes of each node have the average degree equal to k. In his experiments Brin set the minimum of the degree to 2 and the maximum to $\min(5k, 200)$. On the basis of experiments he reported that good balance was not crucial for the performance of the tree.

We have implemented this balancing procedure too. In case of $k = 2$ the value 2 is both the average and the minimal possible value of the splitting degree in the k-means balanced indexing tree so the balancing procedure assigns the degree 2 to all nodes and it behaves identically as in case of the constant degree 2. Hence, the comparison of the balanced and the constant degree selections makes sense for the value of k greater than 2. Figure 17 presents the comparison between the

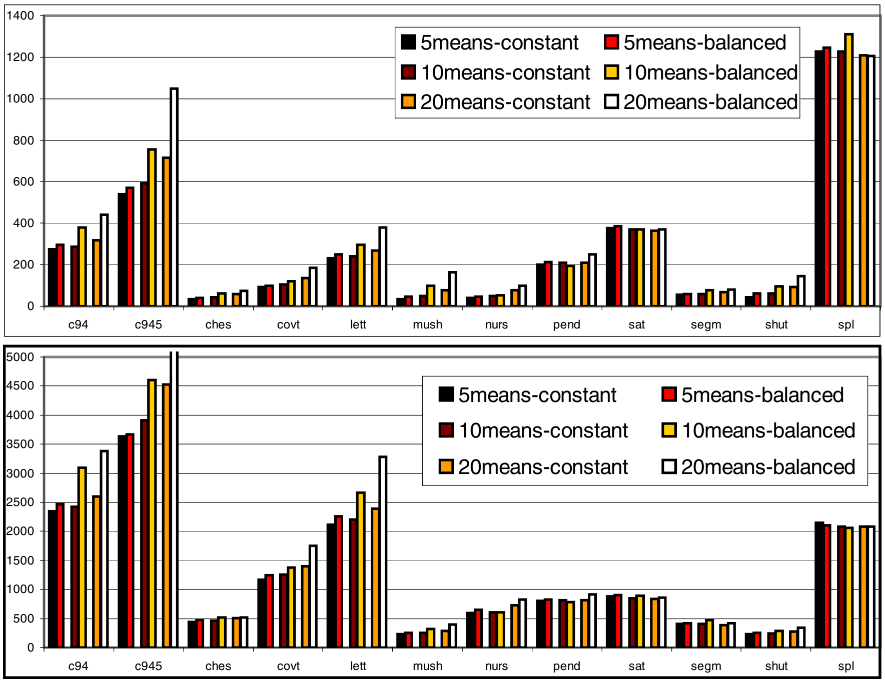


Fig. 17. The average number of distance computations per single object in 1-nn search (the upper graph) and in 100-nn search (the lower graph) with the use of the k-means based indexing trees with constant and with balanced degrees of tree nodes; for each data set the first pair of columns represents the performance for the constant and for the balanced degree $k = 5$, the second pair represents the performance for $k = 10$ and the third one for $k = 20$

k-means based balanced trees where k is the average degree of child nodes and the corresponding k-means trees with the constant degree k. The results show that the balancing procedure does not improve performance of the tree with a constant degree and in many experiments searching in the tree with a constant degree is even faster. It indicates that in order to make profit from balancing more sophisticated procedures are required. Up to now it is not known whether there is a balancing policy with acceptable computational complexity having a significant advantage over the non-balanced structures.

4.5 Searching in the Indexing Tree

In this subsection we present Algorithm 6 that is a general searching schema finding a fixed number k of data objects nearest to the query q [35]. The algorithm traverses the indexing tree rooted at *root* in the depth-first order. In *nearestQueue* it stores the nearest data objects, maximally k, from already visited nodes. At each tree node n the algorithm checks with pruning criteria

Algorithm 6. Searching schema

```
$root$ - the root node of the indexing tree to be searched
$nodeStack$ - the stack of nodes to be searched
$nearestQueue$ - the queue of the data objects nearest to $q$
                 sorted according to the distance $\rho$
$discard(n : node, q : query, r_q : range)$ - the procedure checking whether
             pruning criteria apply to a node $n$ while searching
             the neighbors of $q$ in the distance less or equal $r_q$

$nodeStack := \{root\}$
repeat
   $n :=$ pull the top node from $nodeStack$
   $r_q := \max_{x \in nearestQueue} \rho(q, x)$
   if $|nearestQueue| < k$ or not $discard(n, q, r_q)$
      if $n$ is a leaf
         for each data object $x \in n$
            if $|nearestQueue| < k$ then add $x$ to $nearestQueue$
            else
               check $x$ against the farthest
               object $y \in nearestQueue$
               and replace $y$ with $x$ if $\rho(q, x) < \rho(q, y)$
      else
         push the child nodes of $n$ to $nodeStack$ in the decreasing
         order of the distance of the the child node centers
         to the query $q$ (the nearest on the top)
until $nodeStack$ is empty
return $nearestQueue$
```

whether n should be visited, i.e., whether n can contain an object that is closer to the query q than any previously found nearest neighbor from *nearestQueue*. If so and the node n is a leaf, it compares each data object $x \in n$ against data objects in *nearestQueue* and replaces the farthest object y from *nearestQueue* by x, if x is closer to the query q than y. In case where the node n is an inner node it adds the child nodes of n to *nodeStack* to be visited in the future.

The important issue for efficiency of the algorithm is the selection of a heuristic procedure determining the order of visiting child nodes. The child nodes of the same parent node are visited always in the increasing order of the distance between the center of a child node and the query q, i.e., the child node with the nearest center is visited first and the child node with the farthest center is visited last. The closer center of a child node is to the query q the closer objects to the query are contained in this node. If the nodes with the nearest centers are visited first, it is more probable to find near neighbors quickly and to limit the range of search r_q to a small radius. Thus, more nodes are discarded during further search.

In Subsection 4.6 different node pruning criteria for the function *discard* are described and compared.

4.6 Optimization of Searching in the Indexing Tree

Algorithm 6 presents the searching procedure that uses search pruning criteria to discard nodes while traversing the indexing tree. If the algorithm finds the first k objects and inserts them to *nearestQueue* it starts to check with the procedure $discard(n, q, r_q)$ whether subsequent visited nodes can contain objects closer to the query q than any previously found neighbor from *nearestQueue*. The algorithm does it in the following way: it stores the current search radius r_q defined as the distance $\rho(q, y)$ between the query q and the farthest from q object $y \in nearestQueue$ and for each visited node it checks whether the node can contain an object x such that $\rho(q, x) < r_q$.

The definition of the k nearest neighbor classification model from Subsection 2.5 does not use the axioms of metric from Subsection 2.3. Those axioms are not required for the model but they serve two other purposes. In the first place, they represent mathematically the natural properties of the notion of analogy. Second, all the metric axioms are necessary for correctness of search pruning rules described in the literature [18, 20, 46, 78, 86]. In this subsection we describe all these pruning rules and we propose a combination of the presented rules into one rule.

The most common search pruning criterion applied in BST [46], SS-tree [86] and M-tree [20] uses the covering radius (Figure 18a). Each node n of the indexing tree keeps the center c_n computed with the function $getCenter(n)$ and the covering radius r_n:

$$r_n := \max_{x \in n} \rho(c_n, x).$$

A node n is discarded from searching if the intersection between the ball around q containing all nearest neighbors from *nearestQueue* and the ball containing all members of the node n is empty:

$$\rho(c_n, q) > r_q + r_n.$$

Uhlmann has proposed another criterion for his Generalized-Hyperplane Tree (GHT) [78]. The important assumption for correctness of this criterion is that at the end of the splitting procedure (see Algorithm 4) each object from a parent node is assigned to the child node with the nearest center. It is ensured with the stopping condition: the splitting procedure stops if the centers from the last and the last but one iteration are the same. The procedure returns the object assignment to the centers from the last but one iteration and this stopping condition makes this assignment appropriate for the final centers too.

Uhlmann's criterion uses the hyperplanes separating the child nodes of the same parent (Figure 18b). A node n_i is discarded if there is a brother node n_j of n_i (another child node of the same parent node as n_i) such that the whole query ball is placed beyond the hyperplane separating n_i and n_j (midperpendicular to the segment connecting the centers c_{n_i} and c_{n_j}) on the side of the brother node n_j:

$$\rho(c_{n_i}, q) - r_q > \rho(c_{n_j}, q) + r_q.$$

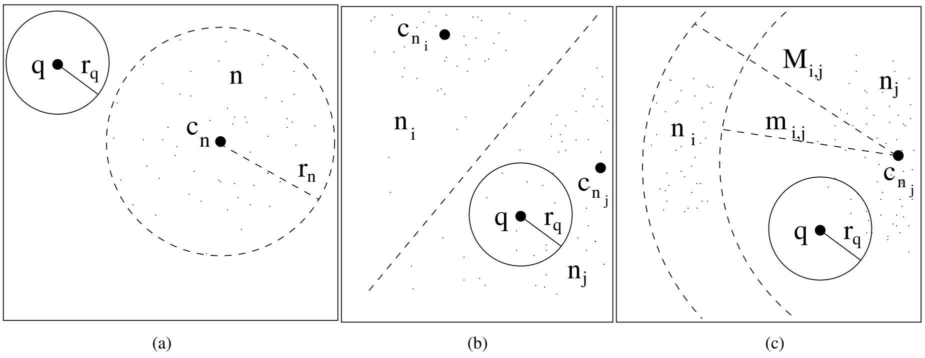


Fig. 18. The three search pruning criteria: (a) the covering radius from BST (b) the hyperplane cut from GHT (c) the rings-based from GNAT

The third pruning criterion used in Brin's GNAT tree [18] is also based on a mutual relation among brother nodes but it is more complex (Figure 18c). If the degree of a tree node is k then each child node n_i keeps the minimum $m_{i,1}, \ldots, m_{i,k}$ and the maximum $M_{i,1}, \ldots, M_{i,k}$ distances between its elements and the centers $c_{n_1}, \ldots, c_{n_k}$ of the remaining brother nodes:

$$m_{i,j} = \min_{x \in n_i} \rho(c_{n_j}, x),$$
$$M_{i,j} = \max_{x \in n_i} \rho(c_{n_j}, x).$$

The node n_i is discarded if there is a brother node n_j such that the query ball is entirely placed outside the ring around the center of n_j containing all members of n_i:

$$\text{either } \rho(c_{n_j}, q) + r_q < m_{i,j} \text{ or } \rho(c_{n_j}, q) - r_q > M_{i,j}.$$

The covering radius and the hyperplane criterion require only to store the center c_n and the covering radius r_n in each node n. The criterion based on the rings requires more memory: each node stores the $2(k-1)$ distances to the centers of the brother nodes.

All the three described criteria are based on the notion of the center of a node. The hyperplane based criterion requires moreover the object assignment condition to be satisfied but it is ensured with the stopping condition of the splitting procedure. Hence, all the three criteria can be applied simultaneously to the indexing structure described in Subsection 4.2, and in this dissertation we propose their combination as the complex criterion for acceleration of the nearest neighbors search.

Figure 19 presents the experimental comparison of the performance for all possible combinations of the three criteria. In a single form the most effective criterion is the covering radius, the least effective is the hyperplane criterion and the differences in performance among all three criteria are significant. In case of the 100-nn search the covering radius alone is almost as powerful as all the three criteria. Addition of the two remaining criteria does not increase

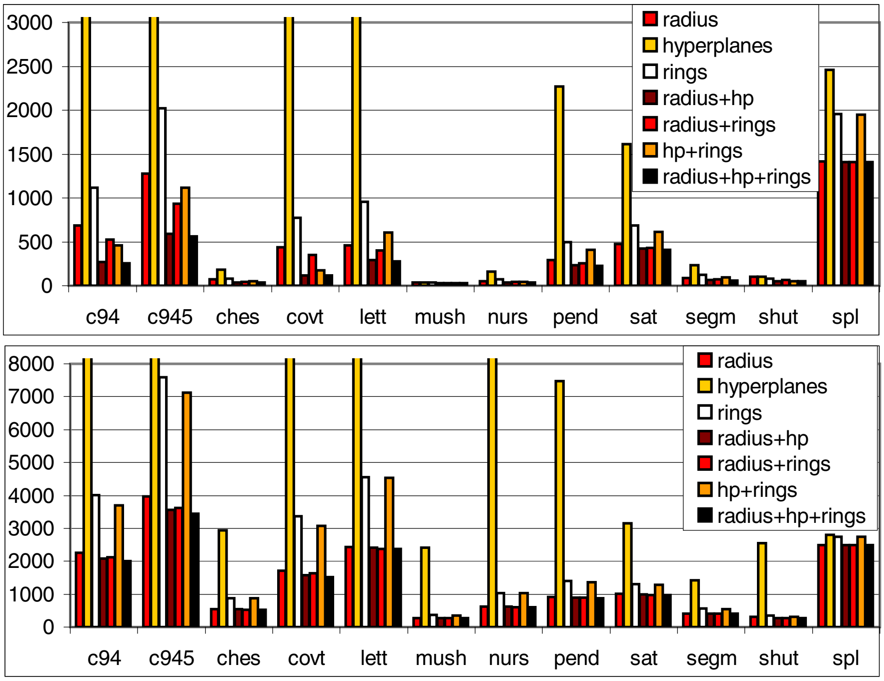


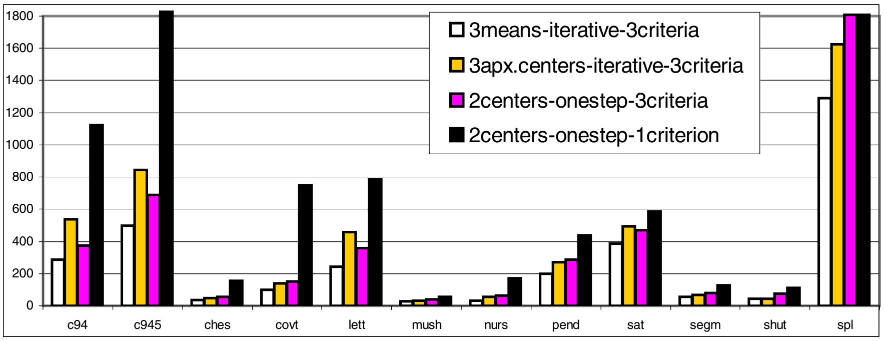
Fig. 19. The average number of distance computations per single object in 1-nn search (the upper graph) and 100-nn search (the lower graph) with the use of the 2-means based indexing tree, and with all the possible combinations of the three search pruning criteria: The covering radius, the hyperplanes and the rings

the performance. The different behavior is observed in case of the 1-nn search: none of them is comparable to the case where all the three criteria are applied. Both the covering radius and the hyperplane cut are crucial for the performance and only the rings based criterion can be removed with no significance loss in the performance.

The presented results indicate that the combination of the different criteria improves the performance of the k-nn search with a single criterion at least for small values of k. On the other hand, in both cases of the 1-nn and the 100-nn search addition of the memory consuming criterion based on rings does not improve the combination of the two remaining criteria. This result may suggest that the covering radius and the hyperplanes provide the optimal pruning combination and there is no need to search for a more sophisticated pruning mechanism.

4.7 Analysis of Searching Cost in the Indexing Tree

The most interesting question is how much the search process profits from the additional cost due to the iterative splitting procedure presented in Algorithm



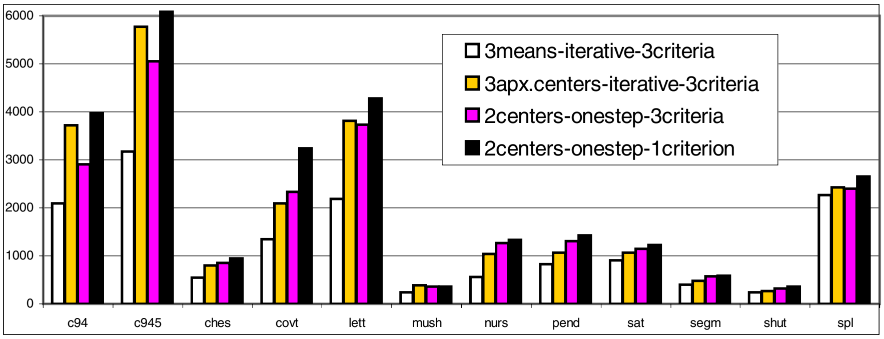


Fig. 20. The average number of distance computations per single object in 1-nn search (the upper graph) and 100-nn search (the lower graph) with the use of the indexing trees with the iterative 3-means, with the iterative 3-approximate-centers, and with the one-step 2-centers splitting procedure, this last tree in two search variants: With the combination of the 3 search pruning criteria and with the single covering radius criterion

4 and the combined search pruning criterion from the previous subsection in comparison to the case with the one-step procedure and a single pruning criterion. The iterative procedure selects initial centers, assigns the data objects to be split to the nearest centers and computes new centers of clusters. Then, the assignment of the data objects to the centers and computation of the new cluster centers is iterated as long as the same set of the cluster centers is generated in two subsequent iterations. The one-step procedure works as in the other indexing trees BST, GHT, GNAT, SS-tree and M-tree. It stops after the first iteration and uses the initial centers as the final ones. The globally farthest data objects are used as the set of the initial centers both in the iterative and in the non-iterative splitting procedure.

Figure 20 presents the cost of searching in the trees with the iterative k-means, with the iterative k-approximate-centers and with the one-step k-centers splitting procedure. The results both for the trees with the iterative procedures and for the first tree with the one-step procedure are obtained with the use of the combination of all the three search pruning criteria. The fourth column at each data set presents the performance of the one-step based tree with the single covering radius criterion. We chose this criterion for comparison since it had the best performance among all the three tested criteria (see Subsection 4.6). Except for a single case we have observed that the performance of the one-step based trees deteriorates while increasing k (it has been checked for $k = 2, 3, 5$ and 7). Then for comparison we have selected the most competitive value $k = 2$ (the exception was the 100-nn search in the data set *splice,* the case $k = 5$ has provided the best performance, and hence, this case has been presented at the graph instead of $k = 2$). In case of both iterative procedures the value $k = 3$ was used since it is one of the most optimal values (see Subsection 4.4).

While comparing the performance of the iterative 3-means (the first column) and the one-step 2-centers (the third column) procedures the profit from applying the iterative procedure is noticeable. In case of the 1-nn search the savings range from 20% (*satimage*) to 50% (*nursery*), in case of the 100-nn search the savings are similar to the 1-nn case, except for a single data set *splice* where the saving is 5%. These results indicate that replacing the one-step procedure with the iterative 3-means procedure can improve the performance even twice.

The comparison between the third and the fourth column presents the profit for the tree with the one-step procedure only from the application of the combined search pruning criterion instead of the single one. For the 1-nn search the combined criterion outperforms significantly the single one, in particular for the largest data sets (*census94, census94-95, covertype*) the acceleration reaches up to several times. For the 100-nn search the difference is not so large but it is still noticeable. These results show that for the tree with the one-step splitting procedure the complex criterion is crucial for the performance of the tree. In case of the tree with the k-means splitting procedure the results are different, i.e., the difference in performance between the single covering radius and the combined criteria is much smaller (see Subsection 4.6). It indicates that the iterative k-means procedure has very good splitting properties and the choice of the search pruning criterion for this case is not so crucial as for the non-iterative case.

The comparison between the first and the fourth columns shows that the tree with the 3-means splitting procedure and the complex search pruning criterion is always at least several tens percent more effective than the tree with the one-step procedure and a single criterion. In case of the 1-nn search the former tree is usually even several times more effective than the latter one.

We obtain different conclusions while comparing the iterative k-approximate-centers (the second column) and the one-step (the third column) procedures. Although for most of data sets the iterative procedure outperforms the non-iterative one, the differences in the performance are usually insignificant and for the three large data sets (*census94, census94-95, letter*) the performance of the iterative procedure is even worse than the performance of the non-iterative one. These results indicate that in case of the tree with the k-approximate-centers the profit in the performance is mainly due to the complex search criterion. Since the only feature that differentiates the k-means and the k-approximate-centers procedures is how the algorithm selects and represents the center of a cluster of data objects this feature seems to be an important issue for the performance of indexing trees.

Uhlmann has introduced another type of an indexing structure: the vantage point tree [78]. It is the binary tree constructed in such a way that at each node the data objects are split with the use of the spherical cut. Given a node n the splitting algorithm picks an object $p \in n$, called the vantage point, and computes the median radius M, i.e., half of the data objects from n fall inside the ball centered at the vantage point p with the radius M and half of them fall outside this ball. The objects inside the ball $\{x \in n : \rho(p,x) \leq M\}$ are inserted into the left branch of the node n and the objects outside the ball $\{x \in n : \rho(p,x) > M\}$ are inserted into the right branch. The vantage point tree is balanced and the construction takes $O(n \log n)$ time in the worst case. While searching for the nearest neighbors of a query q the branch with objects inside the ball is pruned if $M + \rho(q, x_{nearest}) < \rho(p,q)$ and the branch with the objects outside the ball is pruned if $M - \rho(q, x_{nearest}) > \rho(p,q)$. Yianilos has described an implementation of the vantage point tree with sampled selection of the vantage point [95]. For the experimental comparison we have implemented this structure as described by Yianilos.

Figure 21 presents the comparison of the performance of the trees with the 2-means, with the 2-approximate-centers and with the vantage point splitting procedure (since the vantage point tree is a binary tree we use $k = 2$ in all the tested trees to make them comparable). The result are presented only

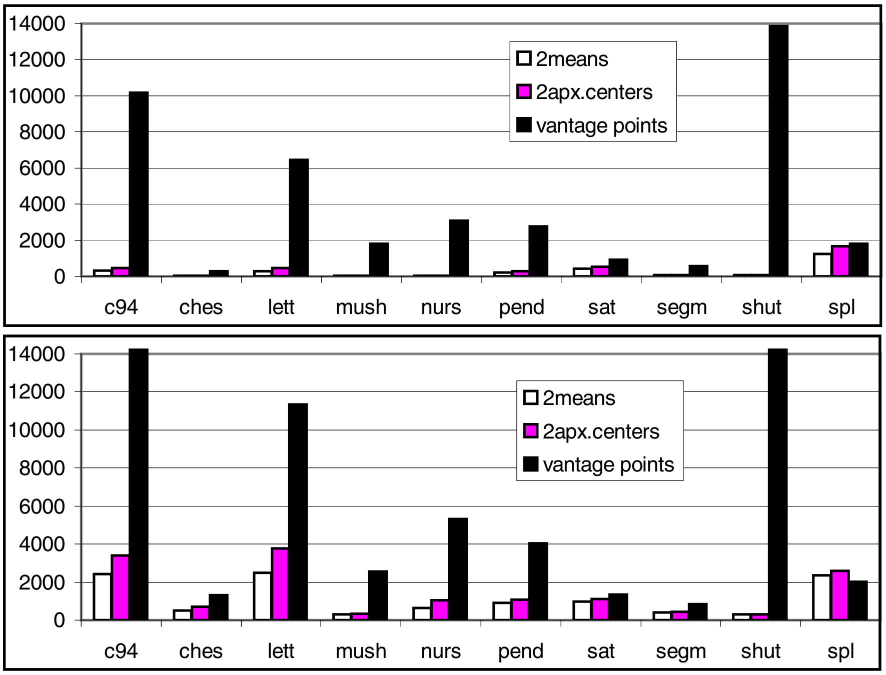


Fig. 21. The average number of distance computations in 1-nn search (the upper graph) and 100-nn search (the lower graph) with the use of the indexing trees with the k-means, with the k-approximate-centers and with the vantage point splitting procedure

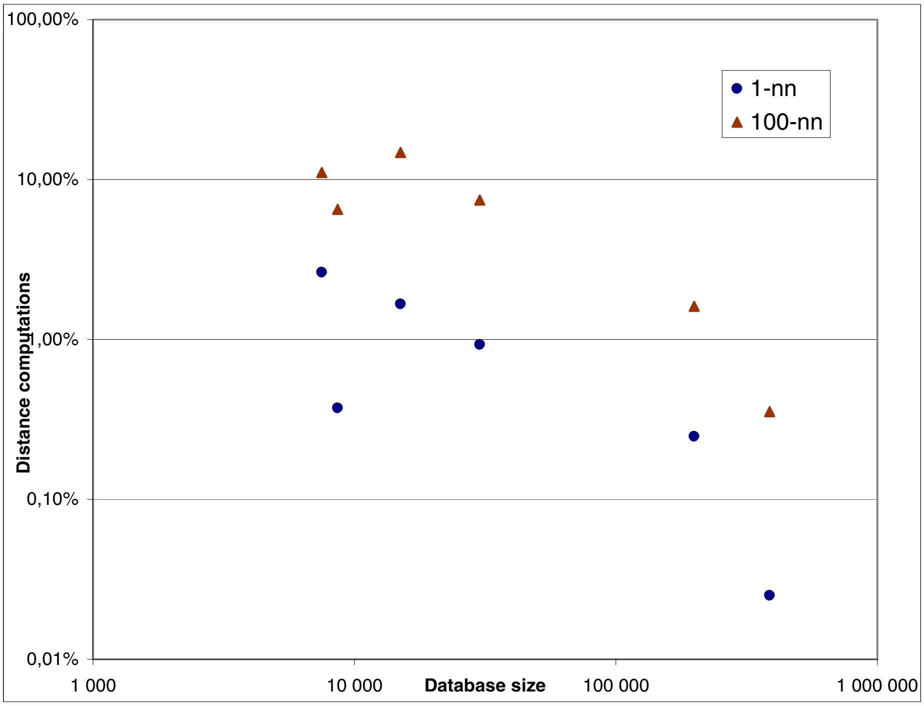


Fig. 22. The average reduction of the number of distance computations in relation to linear search obtained by the 3-means based indexing tree with the 3 search pruning criteria presented in percentage terms in dependence on the training set size

for 10 data sets because for the 2 largest data sets: *census94-95* and *covertype* the experiments with the vantage point tree takes too much time. The results show a large advantage of the trees based on the centers over the tree based on the vantage points. It indicates that the center based representation of tree nodes provides better search pruning properties than the vantage point based representation.

An interesting question is how much of the searching cost the indexing tree with the iterative splitting procedure and the three pruning criteria reduces in relation to the linear scan of a training set. Figure 22 presents the average reduction of the linear search cost for training sets of the different sizes (for the six largest data sets from all 12 sets in Tables 1 and 2). A particularly advanced acceleration level has been reached for the two largest data sets. The size of the data set *covertype* is almost 400 thousand, whereas the average number of distance comparisons per single object (the fourth column set from the left at Figure 20) is less than 100 for the 1-nn search and close to 1300 for the 100-nn search. It means that the 3-means based tree reduces the cost of searching 4000 times in case of the 1-nn search and 300 times in case of the 100-nn search. For the second largest data set *census94-95* (the second column set from the left at Figure 20, the size almost 200 thousand) the reductions in cost are 400 times and 60 times, respectively. This good performance has been reached both due to the improved splitting procedure and due to the use of the complex search criterion.

4.8 Comparison of Searching and Indexing Cost

The results from the previous subsection have proved that the k-means based indexing tree is a good accelerator of searching for the nearest neighbors. The question arises whether the cost of constructing a tree is not too large in comparison to the cost of searching.

Figure 23 presents the comparison between the number of computed distances per single object in the indexing process (in other words the average cost of indexing a single object) and the average number of the distances computed in the 100-nn search.

The results for the k-means and for the k-approximate centers procedure are similar. For $k = 2$ they are quite optimistic, for all the tested data sets except *shuttle* the average cost of indexing a single object is several times lower than the average cost of searching for the 100 nearest neighbors of a single object.

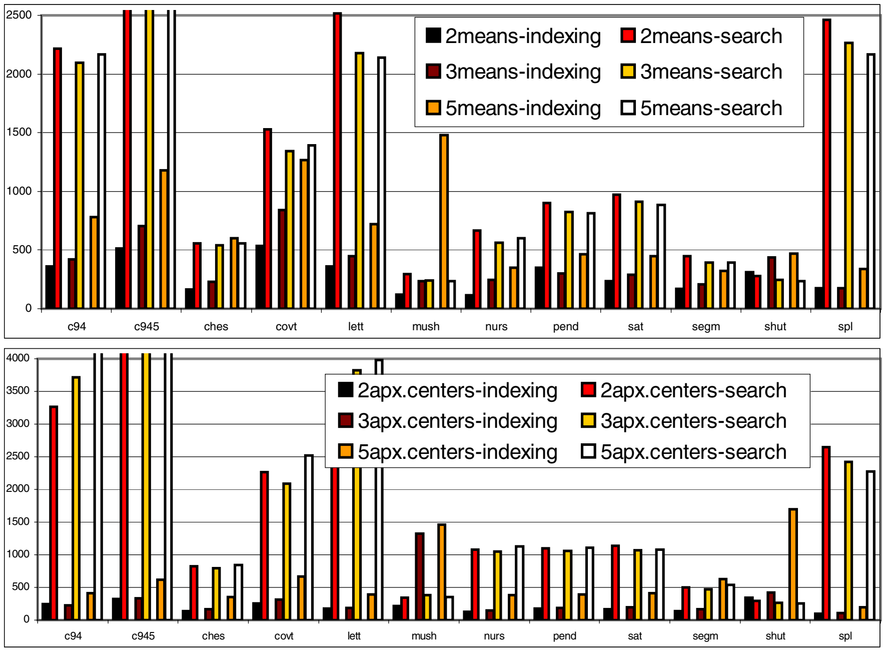


Fig. 23. The average number of distance computations per single object in the indexing algorithm and in 100-nn search, with the use of the indexing trees with the k-means (the upper graph) and with the k-approximate-centers (the lower graph) splitting procedure. For each data set the first pair of columns represents the costs of indexing and searching for $k = 2$, the second pair represents these costs for $k = 3$ and the third one for $k = 5$.

It means that if the size of the training and the test set are of the same order the main workload remains on the side of the search process. For the data sets *shuttle* and *mushroom* the differences in the cost are smaller but it results from the fact that the search process is more effective for these two data sets than for the others.

The situation changes to worse while increasing the degree k. In case of $k = 5$ the cost of indexing for the five data sets: *chess, covertype, mushroom, segment* and *shuttle* is at least comparable and is sometimes higher than the cost of searching. It has been mentioned in Subsection 4.4 that the computational cost of searching is stable for $k \geq 3$. On the other hand, the cost of indexing increases significantly while increasing the degree k. It means that the larger degree k the lower profit from applying the advanced indexing structure is. The results from this subsection and Subsection 4.4 indicate that the best trade-off between the indexing cost and the search performance is obtained for $k = 3$. Increasing the value of k more increases the cost of indexing with no profit from searching.

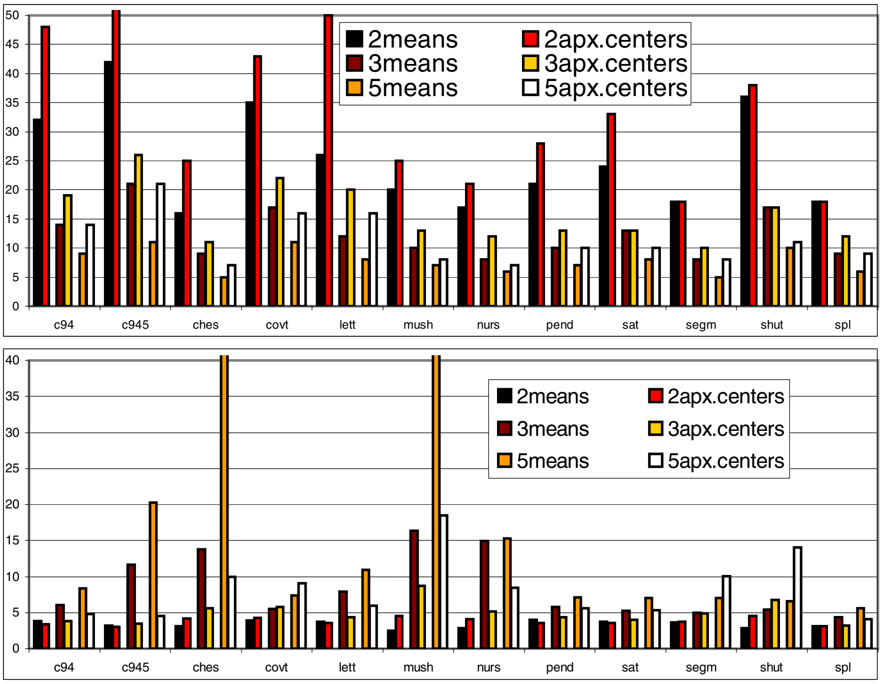


Fig. 24. The height of the tree (the upper graph) and the average number of iterations in the splitting procedure (the lower graph) in the k-means based and in the k-approximate-centers based indexing tree. For each data set the first pair of columns represents the height (at the upper graph) and the iterations (at the lower graph) for the 2-means based and the 2-approximate-centers based tree, the second pair represents these quantities for $k = 3$ and the third one for $k = 5$.

We have analyzed the case of the 100-nn search. In many application, e.g., while searching for the optimal value of k or for some topological properties in a data set, there is the need to search for a large number of the nearest neighbors. In this case the presented trees keep the appropriate balance between the costs of construction and searching. The results for the 1-nn case do not lead to such an unequivocal conclusion. The usefulness of the presented structures for queries with a small k depends more on the specific properties of a data set and on the number of queries to be performed.

The upper graph at Figure 24 provides some information about the shape of the indexing trees. The fact that the height of the trees, i.e., the distance between the root and the deepest leaf, exceeds rarely 25, indicates that the shape of the trees is quite balanced: they do not contain very long thin branches. The lower graph presents the average number of iterations in the splitting procedures. In many experiments, especially for $k = 2$, this number does not exceed 5 what indicates that the construction cost in case of the tree with the iterative splitting procedure is only a few times larger than in case of the tree with the non-iterative procedure.

4.9 Summary

In this section we analyze the different properties of distance based indexing algorithms with center based splitting procedures and search for the optimal parameters of the indexing algorithm. As the result we introduce the following new methods to be used in construction of an indexing structure and in the search process:

– the iterative procedure for splitting nodes in an indexing tree (Subsection 4.2) generalizing the k-means algorithm; the procedure has been presented in two versions: the specific case where the weighted joint city-block and VDM metric is used and the general case of any metric,
– the method for selection of the initial centers in the node splitting procedure (Subsection 4.3); in this method, as distinguished from Brin's method [18], the initial centers are searched globally among all the objects from a given tree node instead of in a sample,
– a complex search pruning criterion combining three different single criteria (Subsection 4.6).

We have compared the three methods for selection of the initial centers in the iterative splitting procedure: random, sampled and global methods. Savaresi and Boley have reported that the 2-means algorithm has good convergence properties [70] so the selection of the initial centers is not very important. This result has been confirmed by the experimental results for most of the tested data sets. However, it was obtained for an infinite theoretical model and there are real-life data sets where the global selection of the initial centers gives a little better performance than the other two methods. We have also observed that the performance of the indexing trees of different splitting degrees is comparable except for the tree of the degree $k = 2$ that has a little worse performance than the trees of the degrees $k \geq 3$. On the other hand, the cost of indexing increases significantly while increasing the degree k. These observations lead to the conclusion that the degree $k = 3$ is the optimal trade-off between the performance of the search process and the cost of indexing.

We have compared the significance of the three different search pruning criteria using the center based representation of tree nodes. Two of criteria are based on the covering radius and on the separating hyperplanes, and the third criterion is based on rings that require more information to be stored at the tree nodes. The experimental results indicate that the most effective criterion is the covering radius. In searching for the 100 nearest neighbors this single criterion is as efficient as all the three criteria combined together. In the case of the 1-nn search none of the tree criteria alone is comparable to all the three criteria used simultaneously, but the combination of the two: the covering radius and the hyperplane criterion is. These results indicate that the two simple criteria define the optimal combination and there is no need to search for a more sophisticated mechanism like the rings based criterion.

The center based indexing trees outperform the vantage point trees. However, the performance of the center based tree still depends much on how the center of a set of data objects is constructed or selected. While comparing the iterative k-means algorithm to the non-iterative one the advantage of the former one is noticeable but the k-means algorithm is applicable only to vector spaces. As a general solution we have proposed the approximate centers that replace the means by centers selected from a sample of objects. Although there is some evidence that the approximate centers perform a little better than the non-iterative centers the difference does not seem to be significant. The gap between the performance of the means and the approximate centers is much larger. These observations indicate that the representation of the center of a set of data objects is crucial for the effectiveness of the center based search pruning and we find the problem of the center selection an important issue for future research.

The experimental results show that the tree with the iterative 3-means splitting procedure and the combined search pruning criteria is up to several times more effective than the one-step based tree with a single criterion. A particularly advanced acceleration level in comparison to the linear search has been reached in case of the largest data sets. The presented structure has reduced the 1-nn search cost 4000 times in case of the data set *covertype* and 400 times in case of the data set *census94-95*. During the 100-nn search the reductions of the performance cost are 300 and 60 times, respectively. These results show the great capability of k-nn based methods in applications to large databases.

It is known that bottom-up constructions give a very good performance but such an immediate construction requires $O(n^2)$ time. Brin, in conclusions of [18], has considered the iterative transformation of the tree from a top-down construction to a bottom-up construction in such a way that at each iteration the tree is constructed with the use of the structure from the previous iteration. Such an approach can result in an indexing structure that reflects more topological properties of a data set than a tree constructed by the top-down method. We find it interesting to instantiate this idea.

The presented indexing and searching method is also described in [91, 92]. It was implemented with the programming language Java and it is used to accelerate the k nearest neighbors classifier in the system RSES [8, 73].

5 Neighborhood-Based Classification Methods

Neighborhood-based classification methods are investigated in this section.

5.1 Estimating the Optimal Neighborhood Size

In the experiments we noticed that the accuracy of the k-nn classifier depends significantly on the number k and different k are appropriate for different data sets. Therefore it is important to estimate the optimal value of k before classification and in this subsection we consider this problem.

Since the optimal value k depends on the data set, we present an algorithm that estimates this optimal value from a training set. The idea is that the leave-one-out classification is applied to the training set in the range of values $1 \leq k \leq k_{max}$ and k with the best leave-one-out accuracy is chosen to be used for a test set. Applying it directly requires repeating the leave-one-out estimation k_{max} times. However, we emulate this process in time comparable to the single leave-one-out test for k equal to the maximum possible value $k = k_{max}$. Algorithm 7 implements this idea.

The function $getClassificationVector(x, k_{max})$ returns the decision of the k-nn classifier for a given object x for the subsequent values of k in the range $1 \leq k \leq k_{max}$. After calling this function for all training objects $x \in U_{trn}$ the algorithm compares the total accuracy of different values k for the whole set U_{trn} and it selects the value k with the maximum accuracy.

At the beginning the function $getClassificationVector(x, k_{max})$ finds the k_{max} training objects from $U_{trn} \setminus \{x\}$ that are nearest to the object x. This is the most time-consuming operation in this function and performing it once instead of for each value $1 \leq k \leq k_{max}$ saves a significant amount of performance time. Then the function counts the votes for particular decisions for successive values of k and for each k it stores the most frequent decision as the result of the classification of the object x. In this way it implements the majority voting model but by analogy one can implement other voting models of the k-nn classifier.

To find the k_{max} nearest training objects from $U_{trn} \setminus \{x\}$ one can use the indexing and searching method described in Section 4 with the small modification: the searching algorithm ignores the object x during search and it does not add it to the set of the nearest neighbors.

The setting $k_{max} = 100$ makes the algorithm efficient enough to apply it to large data sets and we use this setting in further experiments. The maximum

Algorithm 7. The function *findOptimalK* estimating the optimal value k from a training set U_{trn} in the range $1 \leq k \leq k_{max}$

```
function $findOptimalK(k_{max})$
   for each $x \in U_{trn}$
      $A_x := getClassificationVector(x, k_{max})$
   return $\arg\max_{1 \leq k \leq k_{max}} |\{x \in U_{trn} : A_x[k] = dec(x)\}|$

function $getClassificationVector(x, k_{max})$
   $n_1, \ldots, n_{k_{max}}$ := the sequence of the $k_{max}$ nearest neighbors of $x$
            sorted in the increasing order of the distance to $x$
   for each $d_j \in V_{dec}$ $votes[d_j] := 0$
   $mostFrequentDec := \arg\max_{d_j \in V_{dec}} |\{x \in U_{trn} : dec(x) = d_j\}|$
   for $k := 1$ to $k_{max}$
      $votes[dec(n_k)] := votes[dec(n_k)] + 1$
      if $votes[dec(n_k)] > votes[mostFrequentDec]$
            then $mostFrequentDec := dec(n_k)$
      $A_x[k] := mostFrequentDec$
   return $A_x$
```

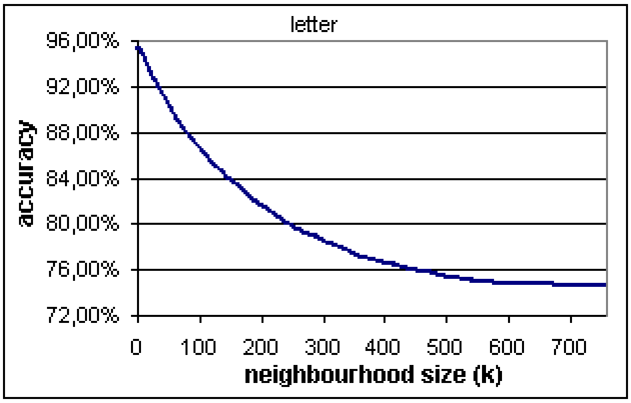
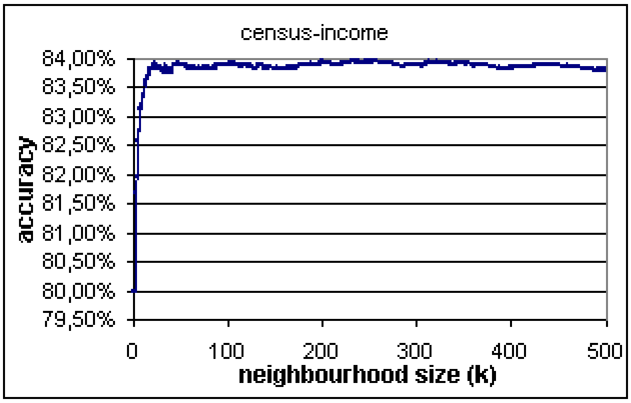


Fig. 25. The classification accuracy for the data set *letter* in dependence on the parameter k

possible value of k_{max} is the size of the training set $|U_{trn}|$. The interesting question is how much the setting $k_{max} = 100$ affects the classification results. To answer this question the following experiment was performed for the data sets from Table 1: for the smallest two data sets: *chess* and *splice* the k-nn accuracy was computed for all possible values of k and for the 8 remaining data sets accuracy was computed for all values k with the maximum value $k_{max} = 500$. For each data set the classification accuracy was measured with the leave-one-out method applied to the training set.



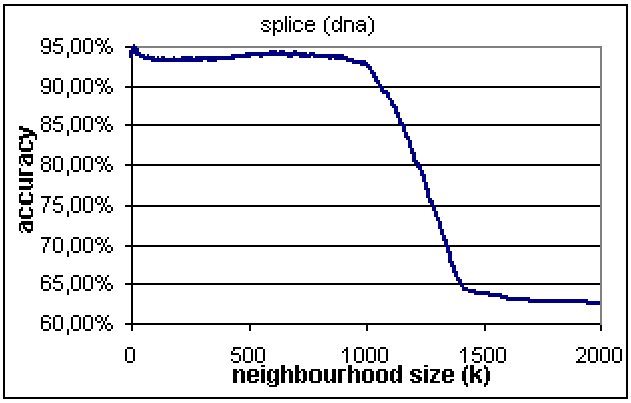


Fig. 26. The classification accuracy for the data sets *census94* and *splice* in dependence on the parameter k

For 8 of the tested data sets (all the sets except *census94* and *splice*) the maximum accuracy was obtained for small values of k (always ≤ 5) and while increasing k the accuracy was significantly falling down (see, e.g., Figure 25).

The dependence between the accuracy and the value k for the two remaining data sets is presented at Figure 26. For the data set *splice* the accuracy remains stable in a wide range of k, at least for the whole range $1 \leq k \leq 1000$, and it starts to fall down for $k > 1000$. However, the maximum accuracy was obtained at the beginning of this wide range: for $k = 15$. In case of the data set *census94* accuracy becomes stable for $k \geq 20$ and it remains stable to the maximum tested value $k = 500$. We observed that the maximum accuracy was obtained for $k = 256$ but the difference to the accuracy for the best k in the range $1 \leq k \leq 100$ was insignificant: accuracy for $k = 24$ was only 0.04% lower than for $k = 256$.

The conclusion is that for all the tested data sets the accuracy reaches a value close to the maximum for a certain small value k. Then either it starts quickly to fall or it remains stable for a wide range of k, but then the fluctuations in the accuracy are very small. The conclusion is that accuracy close to the maximum can be always found in the range $1 \leq k \leq 100$. Therefore, $k_{max} = 100$ provides a good balance between the optimality of the results and the searching time and we assume this setting in further experiments.

5.2 Voting by k Nearest Neighbors

During the classification of any test object x in the k-nn classifier the k nearest neighbors of x vote for different decisions and the classifier chooses the best decision according to a certain voting model.

The most common majority voting model [31] is given in Equation 4. This model assigns the same weight to each object in the set of the k nearest neighbors $NN(x, k)$.

In the literature there are a number of other voting models that take into consideration the distances from the neighbors to the test object x [27, 72]. It has been argued that for a finite training set U_{trn} the distance weighted voting models can outperform the majority voting model [4, 58, 84, 93, 96].

Dudani [27] proposed the inverse distance weight where the weight of a neighbor vote is inversely proportional to the distance from this neighbor to the test object x. In this way closer neighbors are more important for classification than farther neighbors. In the dissertation we consider the modified version of Dudani's model, the inverse square distance weights:

$$dec_{weighted-knn}(x) := \arg \max_{d_j \in V_{dec}} \sum_{y \in NN(x,k): dec(y)=d_j} \frac{1}{\rho(x, y)^2}. \tag{7}$$

In the above model the weight of any neighbor vote is inversely proportional to the square of the distance from this neighbor to the test object x. It makes the weights more diversified than in Dudani's model.

Empirical comparison of the two voting models: with the equal weights and with the inverse square distance weights is discussed in Subsection 5.5.

5.3 Metric Based Generalization of Lazy Rule Induction

In this subsection we consider another approach to learning from examples based on rule induction. Rule induction is one of the most popular approaches in machine learning [7, 21, 42, 60, 74]. Our goal is to combine the k nearest neighbors method with rule induction.

The k-nn model implements the lazy learning approach [6, 34]. In this approach the model is assumed to be induced at the moment of classification. For the k-nn it is implemented in a natural way because the k nearest neighbors are searched in relation to a test object. In this subsection we consider a lazy approach to rule based classification. Bazan [6] has proposed an effective lazy rule induction algorithm for data with nominal attributes. In this subsection, we make the first step towards combining Bazan's algorithm with the k nearest neighbors method: we extend this algorithm to the case of data with both numerical and nominal attributes. The extension uses the assumption that a linear weighted metric from Equation 6 is provided for data. We show that the proposed extension generalizes Bazan's method.

The main feature of rule based classifiers is the set of rules used for classifying objects.

Definition 6. *A* rule *consists of a premise and a consequent:*

$$a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow dec = d_j.$$

The premise is conjunction of attribute conditions and the consequent indicates a decision value. A rule is said to cover an example $x = (x_1, \ldots, x_n)$, and vice versa, the example x is said to match the rule, if all the attribute conditions in the premise of the rule are satisfied by the object values: $x_{i_1} = v_1, \ldots, x_{i_p} = v_p$. The consequent $dec = d_j$ denotes the decision value that is assigned to an object if it matches the rule.

In rule based classifiers a set of rules is induced from a training set. The important properties of rules are consistency and minimality [74].

Definition 7. *A rule $a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow dec = d_j$ is* consistent *with a training set U_{trn} if for each object $x \in U_{trn}$ matching the rule the decision of the rule is correct, i.e., $dec(x) = d_j$.*

The notion of consistency describes the rules that classify correctly all the covered objects in a given training set.

Definition 8. *A consistent rule $a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow dec = d_j$ is* minimal *in a training set U_{trn} if for each proper subset of conditions occurring in the premise of this rule $C \subset \{a_{i_1} = v_1, \ldots, a_{i_p} = v_p\}$ the rule built from these conditions, i.e., $\bigwedge C \Rightarrow dec = d_j$ is inconsistent with the training set U_{trn}.*

The notion of minimality selects the consistent rules of the minimum length in terms of the number of conditions in the premise of a rule. These rules maximize also the set of covered objects in a training set.

The complete set of all minimal consistent rules has good theoretical properties: it corresponds to the set of all rules generated from all local reducts of a given training set [94]. However, the number of all minimal consistent rules can be exponential in relation both to the number of attributes $|A|$ and to the training set size $|U_{trn}|$ and computing all minimal consistent rules is often infeasible [90]. Therefore many rule induction algorithms are based on a smaller set of rules [7, 42].

However, in the dissertation we consider a rule based classification model that allows us to classify objects on the basis of the set of all minimal consistent rules without computing them explicitly. The decision for each object to be classified is computed using the rules covering the object. Usually in a given set of rules they are not mutually exclusive and more than one rule can cover a test object. Therefore a certain model of voting by rules is applied to resolve conflicts between the covering rules with different decisions.

Algorithm 8. Algorithm $decision_{local-rules}(x)$ classifying a given test object x based on lazy induction of local rules

```
for each $d_j \in V_{dec}$ $support[d_j] := \emptyset$
for each $y \in U_{trn}$
    if $r_{local}(x, y)$ is consistent with $U_{trn}$ then
        $support[dec(y)] := support[dec(y)] \cup \{y\}$
return $\arg\max_{d_j \in V_{dec}} |support[d_j]|$
```

Definition 9. *The* support *of a rule $a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow dec = d_j$ in a training set U_{trn} is the set of all the objects from U_{trn} matching the rule and with the same decision d_j:*

$$support(a_{i_1} = v_1 \wedge \ldots \wedge a_{i_p} = v_p \Rightarrow dec = d_j) = \\ \{x = (x_1, \ldots, x_n) \in U_{trn} : x_{i_1} = v_1 \wedge \ldots \wedge x_{i_p} = v_p \wedge dec(x) = d_j\}.$$

In the dissertation, we focus on the commonly used rule based classification model using the notion of the rule support:

$$dec_{rules}(x, R) := \arg\max_{d_j \in V_{dec}} \left| \bigcup_{\alpha \Rightarrow dec = d_j \in R:\, x \text{ satisfies } \alpha} support(\alpha \Rightarrow dec = d_j) \right|. \tag{8}$$

where R is a given set of rules used by the classifier. This model computes the support set for each rule $r \in R$ covering a test object x and then it select the decision with the greatest total number of the supporting objects.

Algorithm 8 presents Bazan's lazy rule induction algorithm $decision_{local-rules}$ [6] that simulates this rule support based classifier dec_{rules} where R is the complete set of all minimal consistent rules. The algorithm was designed originally only for data with nominal attributes and it is based on the following notion of a local rule:

Definition 10. *The* local rule *for a given pair of a test object x and a training object $y \in U_{trn}$ is the rule $r_{local}(x, y)$ defined by*

$$\bigwedge_{a_i : y_i = x_i} a_i = y_i \Rightarrow dec = dec(y).$$

The conditions in the premise of the local rule $r_{local}(x, y)$ are chosen in such a way that both the test object x and the training object y match the rule and the rule is maximally specific relative to the matching condition. This is opposite to the definition of a minimal consistent rule where the premise of a rule is minimally specific. However, there is the following relation between minimal consistent rules and local rules:

Fact 11. *[6] The premise of a local rule $r_{local}(x, y)$ for a test object x and a training object $y \in U_{trn}$ implies the premise of a certain minimal consistent rule if and only if the local rule $r_{local}(x, y)$ is consistent with the training set U_{trn}.*

It means that if a local rule is consistent with a training set then it can be generalized to a certain minimal consistent rule covering both the test and the training object and this property is used to compute the support set of minimal consistent rules matched by a test object in Algorithm 8. Instead of computing all minimal consistent rules covering a given test object x to be classified the algorithm generates the local rules spanned by the object x and each training object $y \in U_{trn}$, and next, it checks the consistency of each local rule against the training set U_{trn}. If the local rule $r_{local}(x, y)$ is consistent with the training set, then the object y supports a certain minimal consistent rule and the algorithm uses y to vote. Hence, the following conclusion can be drawn:

Corollary 12. *[6] The classification result of the rule support based classifier from Equation 8 with the set R of all minimal consistent rules and the lazy local rule induction classifier (Algorithm 8) is the same for each test object x:*

$$dec_{rules}(x, R) = decision_{local-rules}(x).$$

To check the consistency of a local rule $r_{local}(x, y)$ with the training set U_{trn} the algorithm checks for each object $z \in U_{trn}$ with the decision different from y: $dec(z) \neq dec(y)$ whether z matches the local rule. Hence, the time complexity of the lazy rule induction algorithm for a single test object is $O(|U_{trn}|^2 \, |A|)$ and the classification of the whole test set U_{tst} has the time complexity $O(|U_{trn}|^2 \, |U_{tst}| \, |A|)$. It means that lazy induction of rules reduces the exponential time complexity of the rule based classifier to the polynomial time. This makes it possible to apply this algorithm in practice.

The original version of the algorithm was proposed for data only with nominal attributes and it uses equality as the only form of conditions on attributes in the premise of a rule (see Definition 6). We generalize this approach to data with both nominal and numerical attributes and with a metric ρ defined by linear combination of metrics for particular attributes (see Equation 6). Equality as the condition in the premise of the rule from Definition 6 represents selection of attribute values, in this case always a single value. We replace equality conditions with a more general metric based form of conditions. This form allows us to select more than one attribute value in a single attribute condition, and thus, to obtain more general rules.

First, we define the generalized versions of the notions of rule and consistency.

Definition 13. *A* generalized rule *consists of a premise and a consequent:*

$$\rho_{i_1}(a_{i_1}, v_1) \leq r_1 \wedge \ldots \wedge \rho_{i_p}(a_{i_p}, v_p) < r_p \Rightarrow dec = d_j.$$

Each condition $\rho_{i_j}(a_{i_j}, v_j) \leq r_j$ or $\rho_{i_j}(a_{i_j}, v_j) < r_j$ in the premise of the generalized rule is described as the range of acceptable values of a given attribute a_{i_q} around a given value v_q. The range is specified by the distance function ρ_{i_q} that is the component of the total distance ρ and by the threshold r_q.

The definition of rule consistency with a training set for the generalized rules is analogous to Definition 7.

Definition 14. *A consistent generalized rule* $\rho_{i_1}(a_{i_1}, v_1) < r_1 \wedge \ldots \wedge \rho_{i_p}(a_{i_p}, v_p) < r_p \Rightarrow dec = d_j$ *is* minimal *in a training set* U_{trn} *if for each attribute* $a_{i_q} \in \{a_{i_1}, \ldots, a_{i_p}\}$ *occurring in the premise of the generalized rule the rule* $\rho_{i_1}(a_{i_1}, v_1) < r_1 \wedge \ldots \wedge \rho_{i_q}(a_{i_q}, v_q) \le r_q \wedge \ldots \wedge \rho_{i_p}(a_{i_p}, v_p) < r_p \Rightarrow dec = d_j$ *with the enlarged range of acceptable values on this attribute (obtained by replacing* $<$ *by* $\le$ *in the condition of the original rule) is inconsistent with the training set* U_{trn}.

Observe, that each condition in the premise of a minimal consistent generalized rule is always a strict inequality. It results from the assumption that a training set U_{trn} *is finite.*

For the generalized version of the classifier based on the set of all minimal consistent rules we use the notion of a generalized rule center.

Definition 15. *An object* $(x_1, \ldots, x_n)$ *is* center *of the rule from Definition 13 if for each attribute* a_{i_j} *from its premise we have* $x_{i_j} = v_j$.

Observe, that a rule can have many centers if there are attributes that do not occur in the premise of the rule.

In the generalized rule support based classification model the support is counted using the set R equal to the set of all generalized minimal consistent rules centered at a test object x:

$$decision_{gen-rules}(x, R) := \arg\max_{d_j \in V_{dec}} \left| \bigcup_{\alpha \Rightarrow dec = d_j \in R:\, x \text{ is a center of } \alpha \Rightarrow dec = d_j} support(\alpha \Rightarrow dec = d_j) \right|. \quad (9)$$

Although in the generalized version we consider only minimal consistent rules centered at a test object the number of these rules can be exponential as in the non-generalized version:

Fact 16. *For arbitrary large set of attributes* A *there is a training set* U_{trn} *and a test object* x *such that the number of minimal consistent rules centered at* x *is exponential with respect both to the number of attributes* $|A|$ *and to the size of the training set* $|U_{trn}|$.

Proof. We assume that the number of attributes $n = |A|$ is even and the decision is binary: $V_{dec} = \{0, 1\}$. In the proof we use any linear metric from Equation 6 to define distance between attribute values and we assume only that each attribute has at least two different values, let us assume that $\{0, 1\} \subseteq V_i$. We define the training set U_{trn} consisting of $\frac{n}{2} + 1$ objects. The first object x^0 has all the attribute values and the decision value equal to 0: $x_i^0 = 0$, $dec(x^0) = 0$. Any object x^j from the remaining $\frac{n}{2}$ objects $x^1, \ldots, x^{\frac{n}{2}}$ has the two values of neighboring attributes and the decision value equal to 1: $x_{2j-1}^j = x_{2j}^j = 1$, $dec(x^j) = 1$ and the remaining attributes have the value equal to 0: $x_i^j = 0$ $(i \ne 2j-1, 2j)$. Consider the object $x = x^0$ and minimal consistent rules centered at x. To exclude each of the training object x^j a minimal consistent rule contains the condition with exactly one of the two attributes that have the value 1 in the object x^j. On the other hand the rule can contain the condition with any attribute from each pair a_{2j-1}, a_{2j}. It means the for each selection function $sel : \{1, \ldots, \frac{n}{2}\} \rightarrow \{0, 1\}$ there is the corresponding minimal consistent rule:

$$\bigwedge_{1 \leq j \leq \frac{n}{2}} \rho_{2j-sel(j)}(a_{2j-sel(j)}, x^0_{2j-sel(j)}) < \rho_{2j-sel(j)}(0, 1) \Rightarrow dec = 0.$$

Each of the above rules is unique. Hence, the number of minimal consistent rules centered at x is $2^{\frac{n}{2}}$ and we obtain the following exponential relation between this number of minimal consistent rules denoted by R_x and the number of attributes and the training set size:

$$R_x = 2^{|U_{trn}|-1} = (\sqrt{2})^{|A|}. \qquad \square$$

Since it is impossible to enumerate all generalized minimal consistent rules in practice we propose to simulate the generalized rule support based classification model from Equation 9 by analogy to Algorithm 8. First, we introduce the definition of a generalized local rule analogous to Definition 10. The conditions in a generalized local rule are chosen in such a way that both the test and the training object match the rule and the conditions are maximally specific.

Definition 17. *The* generalized local rule *for a given pair of a test object x and a training object $y \in U_{trn}$ is the rule $r_{gen-local}(x, y)$:*

$$\bigwedge_{a_i \in A} \rho_i(a_i, x_i) \leq \rho_i(y_i, x_i) \Rightarrow dec = dec(y).$$

For each attribute a_i the range of acceptable values in the corresponding condition of the generalized local rule is defined as the set of values whose distance to the attribute value x_i in the test object is less or equal to the distance from the attribute value y_i in the training object to x_i.

First, we identify the relation between the original and the generalized notion of a local rule. Let us consider the case where to define the generalized rules the Hamming metric described in Subsection 2.4 is used for all the attributes, both the nominal and the numerical ones.

Fact 18. *For the Hamming metric the notion of the generalized local rule $r_{gen-local}(x, y)$ in Definition 17 is equivalent to the notion of the local rule $r_{local}(x, y)$ in Definition 10.*

Proof. Consider a single attribute a_i. If the values of x and y on this attribute are equal $x_i = y_i$ the corresponding condition in the local rule $r_{local}(x, y)$ has the form of equality $a_i = y_i$. The attribute distance in the Hamming metric between two equal values is 0 so the corresponding condition in the generalized local rule has the form $\rho_i(a_i, x_i) \leq 0$. The distance between two attribute values in the

Hamming metric is 0 if and only if these two value are equal. Hence, in case of $x_i = y_i$ the corresponding conditions $a_i = y_i$ and $\rho_i(a_i, x_i) \leq 0$ in the local and in the generalized local rule, respectively, are equivalent.

If the values of x and y on the attribute a_i are different $x_i \neq y_i$ the condition corresponding to the attribute a_i does not occur in the local rule $r_{local}(x, y)$. This means that the local rule accepts all values on this attribute. In the generalized local rule $r_{gen-local}(x, y)$ the corresponding condition has the form $\rho_i(a_i, x_i) \leq 1$. But in the Hamming metric the attribute distance between two values is always either 0 or 1 so the condition $\rho_i(a_i, x_i) \leq 1$ is satisfied for all the values of the attribute a_i too.

Hence, for each attribute the corresponding conditions in the local rule $r_{local}(x, y)$ and in the generalized rule $r_{gen-local}(x, y)$ with the Hamming metric are equivalent so the whole premises of these two rules are equivalent too. □

Now we present an example how the presented generalization works for the case of a non-trivial metric. We consider the joint city-block and VDM metric defined in Subsection 3.1. Let us assume that the following training set is provided:

Object	**Age (A)**	**Weight (W)**	**Sex (S)**	**BloodGroup (BG)**	**Diagnosis**
y_1	35	90	M	A	Sick
y_2	40	65	F	AB	Sick
y_3	45	68	F	AB	Healthy
y_4	40	70	M	AB	Healthy
y_5	45	75	M	B	Sick
y_6	35	70	F	B	Healthy
y_7	45	70	M	0	Healthy

Age and *Weight* are the numerical attributes and *Sex* and *BloodGroup* are the nominal attributes. We consider the following test object:

Object	**Age (A)**	**Weight (W)**	**Sex (S)**	**BloodGroup (BG)**	**Diagnosis**
x_1	50	72	F	A	?

For the attribute *BloodGroup* there are 4 possible values: A, AB, B and 0. To construct the generalized local rules for x_1 we need to compute the attribute distance from A to each other value:

$$\begin{aligned}
\rho_{BG}(A, A) &= 0, \\
\rho_{BG}(A, AB) &= |P(Diagn = Healthy|A) - P(Diagn = Healthy|AB)| - \\
&\quad - |P(Diagn = Sick|A) - P(Diagn = Sick|AB)| = \\
&= \left|0 - \frac{2}{3}\right| - \left|1 - \frac{1}{3}\right| = \frac{4}{3}, \\
\rho_{BG}(A, B) &= |P(Diagn = Healthy|A) - P(Diagn = Healthy|B)| - \\
&\quad - |P(Diagn = Sick|A) - P(Diagn = Sick|B)| = \\
&= \left|0 - \frac{1}{2}\right| - \left|1 - \frac{1}{2}\right| = 1,
\end{aligned}$$

$$\begin{aligned}\rho_{BG}(A,0) = &|P(Diagn = Healthy|A) - P(Diagn = Healthy|0)| - \\ &- |P(Diagn = Sick|A) - P(Diagn = Sick|0)| = \\ &= |0-1| - |1-0| = 2.\end{aligned}$$

Consider the generalized local rule $r_{gen-local}(x_1, y_1)$. Since the objects x_1 and $y_1$have the same value A on the attribute *BloodGroup*, the local rule accepts only this value on the attribute *BloodGroup*:

$$A \in [35; 65] \wedge W \in [54; 90] \wedge BG = A \Rightarrow Diagn = Sick.$$

No other training object except for y_1 satisfies the premise of this rule so it is consistent and it can be extended to a minimal consistent rule, e.g.,

$$BG = A \Rightarrow Diagn = Sick.$$

If we consider the generalized local rule $r_{gen-local}(x_1, y_2)$ for the objects x_1 and y_2, the distance between the values of x_1 and y_2 on the attribute *BloodGroup* is $\rho_{BG}(A, AB) = \frac{4}{3}$. It makes the three values A, AB and B be accepted in the rule $r_{gen-local}(x_1, y_2)$ on the attribute *BloodGroup*:

$$A \in [40; 60] \wedge W \in [65; 79] \wedge S = F \wedge BG \in \{A, AB, B\} \Rightarrow Diagn = Sick.$$

Now we obtain the inconsistent rule because, e.g., the object y_3 satisfies the premise of this rule and it has the inconsistent decision $Diagn = Healthy$.

The most important property of the presented generalization is the relation between generalized minimal consistent rules and generalized local rules analogous to Fact 11.

Theorem 19. *The premise of the generalized local rule $r_{gen-local}(x, y)$ for a test object x and a training object $y \in U_{trn}$ implies the premise of a certain generalized minimal consistent rule centered at x if and only if the generalized local rule $r_{local}(x, y)$ is consistent with the training set U_{trn}.*

Proof. First, we show that if the generalized local rule $r_{gen-local}(x, y)$ is consistent with the training set U_{trn} it can be extended to the generalized minimal rule centered at x. We define the sequence of rules $r^0, \ldots, r^n$ in the following way. The first rule in th sequence is the local rule $r^0 = r_{gen-local(x,y)}$. To define each next rule r_i we assume that the previous rule r_{i-1}:

$$\bigwedge_{1 \le j < i} \rho_j(a_j, x_j) < M_j \bigwedge_{i \le j \le n} \rho_j(a_j, x_j) \le \rho_j(y_j, x_j) \Rightarrow dec = dec(y).$$

is consistent with th training set U_{trn} and the first $i-1$ conditions of the rule r_{i-1} are maximally general, i.e., replacing any strong inequality $\rho_j(a_j, x_j) < M_j$ for $j < i$ by the weak makes this rule inconsistent. Let S_i be the set of all the object that satisfy the premise of the rule r_{i-1} with the condition on the attribute a_i removed:

$$S_i = \{z \in U_{trn} : z \text{ satisfies } \bigwedge_{1 \le j < i} \rho_j(a_j, x_j) < M_j \bigwedge_{i < j \le n} \rho_j(a_j, x_j) \le \rho_j(y_j, x_j)\}.$$

In the rule r_i the i-th condition is maximally extended in such way that the rule remains consistent. It means that the range of acceptable values for the attribute a_i in the rule r_i has to be not larger than the attribute distance from x to any object in S_i with a decision different from $dec(y)$. If S_i does not contain an object with a decision different from $dec(y)$ the range remains unlimited:

$$M_i = \begin{cases} \infty & \text{if } \forall z \in S_i \, dec(z) = dec(y) \\ \min\{\rho_i(z_i, x_i) : z \in S_i \wedge dec(z) \neq dec(y)\} & \text{otherwise.} \end{cases} \tag{10}$$

If we limit the range of values on the attribute a_i in the rule r_i by the M_i with the strong inequality in the condition:

$$\bigwedge_{1 \leq j < i} \rho_j(a_j, x_j) < M_j \wedge \rho_i(a_i, x_i) < M_i \bigwedge_{i < j \leq n} \rho_j(a_j, x_j) \leq \rho_j(y_j, x_j) \Rightarrow dec = dec(y)$$

then it ensures that the rule r_i remains consistent. On the other hand, the value of M_i in Equation 10 has been chosen in such a way that replacing the strong inequality by the weak inequality or replacing the range by a value larger than M_i causes the situation where a certain object with a decision different from $dec(y)$ satisfies the condition on the attribute a_i and the whole premise of the rule r_i, i.e., the rule r_i becomes inconsistent.

Since r_{i-1} was consistent the range M_i is greater than the range for the attribute a_i in the rule r_{i-1}: $M_i > \rho(y_i, x_i)$. Hence, the ranges for the previous attributes $M_1, \ldots, M_{i-1}$ remain maximal in the rule r_i: widening of one of these ranges in the rule r_{i-1} makes an inconsistent object match r_{i-1} and the same happens for the rule r_i.

By induction the last rule $r_n : \bigwedge_{1 \leq j \leq n} \rho_j(a_j, x_j) < M_j \Rightarrow dec = dec(y)$ in the defined sequence is consistent too and all the conditions are maximally general. Then r_n is consistent and minimal. Since the premise of each rule r_{i-1} implies the premise of the next rule r_i in the sequence and the relation of implication is transitive the first rule r_0 that is the generalized local rule $r_{gen-local}(x, y)$ of the objects x, y implies the last rule r_n that is a minimal consistent rule. Thus we have proved the theorem for the case when the generalized local rule is consistent.

In case where the generalized local rule $r_{gen-local}(x, y)$ is inconsistent each rule centered at x implied by $r_{gen-local}(x, y)$ covers all objects covered by $r_{gen-local}(x, y)$, in particular it covers an object causing inconsistency. Hence, each rule implied by $r_{gen-local}(x, y)$ is inconsistent too. □

The above theorem allows to define an effective generalized version of the local rule based algorithm simulating the rule support based classifier (see Algorithm 8). Algorithm 9 works in the same way as the non-generalized version. Instead of computing all the generalized minimal consistent rules centered at a given test object x to be classified the algorithm generates the generalized local rules spanned by the object x and each training object $y \in U_{trn}$ and then checks consistency of each local rule against the training set U_{trn}. The time complexity of the generalized lazy rule induction algorithm is the same as the complexity of

Algorithm 9. Algorithm $decision_{gen-local-rules}(x)$ classifying a given test object x based on lazy induction of the generalized local rules

```
for each $d_j \in V_{dec}$ $support[d_j] := \emptyset$
for each $y \in U_{trn}$
    if $r_{gen-local}(x,y)$ is consistent with $U_{trn}$ then
        $support[dec(y)] := support[dec(y)] \cup \{y\}$
return $\arg\max_{d_j \in V_{dec}} |support[d_j]|$
```

the non-generalized version: $O(|U_{trn}|^2 \, |U_{tst}| \, |A|)$. Theorem 19 ensures that the algorithm counts only those objects that are covered by a certain generalized minimal consistent rule centered at x. Hence, we obtain the final conclusion.

Corollary 20. *The classification result of the generalized rule support based classifier from Equation 9 with the set R of all the generalized minimal consistent rules centered at x and the generalized lazy local rule induction classifier (Algorithm 9) is the same for each each test object x:*

$$decision_{gen-rules}(x,R) = decision_{gen-local-rules}(x).$$

In this way we extended the effective lazy rule induction algorithm for data with nominal attributes to the case of data with both nominal and numerical attributes and with linear weighted distance provided.

5.4 Combination of k Nearest Neighbors with Generalized Lazy Rule Induction

In this subsection we consider an approach from multistrategy learning, i.e., a method combining more than one different approaches. We examine the combination of the k nearest neighbors method with rule induction. In the literature there is a number of different solutions combining these two methods [25, 37, 54]. Contrary to the other solutions combining k-nn with rule induction we propose the algorithm that preserves lazy learning, i.e., rules are constructed in lazy way at the moment of classification like the nearest neighbors. The proposed combination uses the metric based generalization of rules described in the previous subsection.

For each test object x Algorithm 9 looks over all the training examples $y \in U_{trn}$ during construction of the support sets $support[d_j]$. Instead of that we can limit the set of the considered examples to the set of the k nearest neighbors of x. The intuition is that training examples far from a test object x are less relevant for classification than closer objects. Therefore in the algorithm combining the two approaches we use the modified definition of the rule support, depending on a test object x:

Definition 21. *The* k-support *of the generalized rule $\alpha \Rightarrow dec = d_j$ for a test object x is the set:*

$$k-support(x, \alpha \Rightarrow dec = d_j) = \{y \in NN(x,k) : y \ matches \ \alpha \wedge dec(x) = d_j\}.$$

The k-support of the rule contains only those objects from the original support set that belong to the set of the k nearest neighbors.

Now, we define the classification model that combines the k-nn method with rule induction by using the k-supports of the rules:

$$decision_{knn-rules}(x,R) := \arg\max_{d_j \in V_{dec}} \left| \bigcup_{r \in R:\, r \text{ centered in } x} k\!-\!support(x,r) \right|. \quad (11)$$

In the above model R is the set of all generalized minimal consistent rules. The difference between the generalized rule support based classifier $decision_{gen-rules}$ from Equation 9 and the combined classifier $decision_{knn-rules}$ is that the combined classifier counts only those objects supporting minimal consistent rules that belong to the set of the k nearest neighbors.

The form of the definition of the combined classifier $decision_{knn-rules}$ in Equation 11 presents the difference between the combined classifier and the pure rule based classifiers described in the previous subsection. Now, we consider the combined classifier from the point of view of the k nearest neighbors method.

Fact 22. *The combined classifier $decision_{knn-rules}$ can be defined by the equivalent formula:*

$$decision_{knn-rules}(x,R) := \arg\max_{d_i \in V_{dec}} \sum_{y \in NN(x,k):\, dec(y)=d_i} \delta(y,R) \quad (12)$$

where the value of $\delta(y)$ is defined by

$$\delta(y) := \begin{cases} 1 \; \textit{if} \; \exists r \in R \; \textit{centered in } x \; \textit{supported by } y \\ 0 \; \textit{otherwise} \end{cases}$$

and R is the set of all generalized minimal consistent rules for the training set U_{trn}.

The above fact shows that the combined classifier presented in this subsection can be considered as a special sort of the k nearest neighbors method: it can be viewed as the k-nn classifier with the specific rule based zero-one voting model. Such a zero-one voting model is a sort of filtering: it excludes some of the k nearest neighbors from voting. Such a voting model can be easily combined with other voting models, e.g., with the inverse square distance weights defined in Equation 7:

$$dec_{weighted-knn-rules}(x,R) := \arg\max_{d_i \in V_{dec}} \sum_{y \in NN(x,k):\, dec(y)=d_i} \frac{\delta(y,R)}{\rho(x,y)^2}. \quad (13)$$

As for the generalized rule support classifier we propose an effective algorithm simulating the combined classifier $decision_{knn-rules}$ based on the generalized local rules. The operation of consistency checking for a single local rule

Algorithm 10. Algorithm $decision_{gen-local-knn-rules}(x)$ simulating the classifier $decision_{knn-rules}(x)$ with lazy induction of the generalized local rules

```
for each $d_j \in V_{dec}$ $support[d_j] := \emptyset$
$neighbor_1, \ldots, neighbor_k :=$ the $k$ nearest neighbors of $x$
                sorted from the nearest to the farthest object
for each $i := 1$ to $k$
   if $r_{gen-local}(x, neighbor_i)$ is consistent
   with $neighbor_1, \ldots, neighbor_{i-1}$ then
      $support[dec(neighbor_i)] := support[dec(neighbor_i)] \cup \{neighbor_i\}$
return $\arg\max_{d_j \in V_{dec}} |support[d_j]|$
```

in Algorithm 9 takes $O(|U_{trn}|\,|A|)$ time. If the linear weighted distance from Equation 6 is used we can use the following fact to accelerate the consistency checking operation in the local rule based algorithm for the combined classifier $decision_{knn-rules}$:

Fact 23. *For each training object $z \in U_{trn}$ matching a generalized local rule $r_{gen-local}(x, y)$ based on a linear weighted distance ρ the distance between the objects x and z is not greater than the distance between the objects x and y:*

$$\rho(x, z) \leq \rho(x, y).$$

Proof. The generalized local rule $r_{gen-local}(x, y)$ for a test object $x = (x_1, \ldots, x_n)$ and a training object $y = (y_1, \ldots, y_n)$ has the form

$$\bigwedge_{a_i \in A} \rho_i(a_i, x_i) \leq \rho_i(y_i, x_i) \Rightarrow dec = dec(y).$$

If $z = (z_1, \ldots, z_n)$ matches the rule then it satisfies the premise of this rule. It means that for each attribute $a_i \in A$ the attribute value z_i satisfies the following condition: $\rho_i(z_i, x_i) \leq \rho_i(y_i, x_i)$. Hence, we obtain that the distance between the objects x and z is not greater than the distance between the objects x and y:

$$\rho(x, z) = \sum_{a_i \in A} w_i \rho_i(z_i, x_i) \leq \sum_{a_i \in A} w_i \rho_i(y_i, x_i) = \rho(x, y). \qquad \square$$

The above fact proves that to check consistency of a local rule $r_{gen-local}(x, y)$ with a training set U_{trn} it is enough to check only those objects from the training set U_{trn} that are closer to x than the object y.

Algorithm 10 presents the lazy algorithm simulating the classifier $decision_{knn-rules}(x, R)$ combining the k nearest neighbors method with rule induction. The algorithm follows the scheme of the generalized local rule based algorithm described in the previous subsection (see Algorithm 9). There are two differences. First, only the k nearest neighbors of a test object x are allowed to vote for decisions. Second, the consistency checking operation for each local rule $r_{gen-local}(x, y)$ checks only those objects from the training set U_{trn} that are closer to x than the object y. Thus the time complexity of the consistency checking operation for a single neighbor is $O(k\,|A|)$. For a single test object the consistency checking operation is performed once for each of the k nearest neighbors. Hence, the cost of consistency checking in the whole procedure testing a single object is $O(k^2\,|A|)$. In practice, it takes less time than searching for the k nearest neighbors. In this way we have obtained an important property of the proposed combination: addition of the rule induction to the k nearest neighbors algorithm does not lengthen significantly the performance time of the k-nn method.

Algorithm 10 simulates the classifier $decision_{knn-rules}(x, R)$ correctly only if the distances from a test object x to training objects are different. To omit this assumption the algorithm requires two small changes. First, the procedure searching for the k nearest neighbors of x returns all objects that are equally distant from x as the k-th nearest neighbor of x. It means that sometimes the algorithm considers more than k nearest neighbors of x. Second, in the procedure checking consistency of a rule $r_{gen-local}(x, neighbor_i)$ the algorithm checks also all the neighbors $neighbor_{i+1}, \ldots, neighbor_{i+l}$ that are equally distant from x as the neighbor $neighbor_i$.

5.5 Experimental Results for Different Voting Models

In this subsection we compare the performance of the k-nn method with different voting models described in the previous subsections. Four voting models are compared: the majority voting model with equal weights defined in Equation 4, the inverse square distance weights (see Equation 7), the zero-one voting model using generalized minimal consistent rules to filter objects (see Equation 12) and the combination of the last two methods, i.e., the inverse square distance weights with rule based filtering (see Equation 13).

On the basis of the results from Section 3 we choose two most effective metrics to be used in the experiments. The first tested metric is the joint city-block and VDM metric (see Subsection 3.1) with the attribute weighting method optimizing distance (see Subsection 3.4) and the second tested metric is the joint DBVDM and VDM metric (see Subsection 3.2) with the same attribute weighting method.

To compare the voting models we performed a number of experiments for the 10 benchmark data sets presented in Table 1. Each data set was partitioned into a training and a test set as described in Subsection 2.7. For each data set and for each voting model the k nearest neighbors method was trained and tested 5 times for the same partition of the data set and the average classification error was calculated for comparison. In each test first the metric was induced from the training set, then the optimal value of k was estimated from the training set in the range $1 \leq k \leq 200$ with Algorithm 7 and finally, the test part of a data set was tested with the k nearest neighbor method for the previously estimated value of k.

Both in case of the joint city-block and VDM metric and in case of the joint DBVDM and VDM metric all the tests for the data set *mushroom* gave the error 0% and all the tests for the data set *shuttle* gave an error not greater than 0.1%.

Since the classification error for these two data sets is always very small it does not provide reliable results to compare different voting models and we focus on the 8 remaining data sets: *census94, chess, letter, nursery, pendigits, satimage, segment* and *splice*.

First, we consider the weighted joint city-block and VDM metric. The table below presents the average value of the estimation of the optimal k for particular voting models with this metric:

Data set	equal weights	sqr. inv. dist. weights	equal weights & rule based filter.	sqr. inv. dist. weights & rule based filter.
census94	30.6	119.4	88.8	181
chess	1	1	1.4	49.2
letter	1	4	1	5.6
nursery	5.4	15.4	19.4	16.8
pendigits	2.2	3.6	2.2	4.2
satimage	2	5.2	2	5.6
segment	1	2.4	1	5.6
splice	1.8	3	2.6	3.6

The estimation of the optimal k for the models with the inverse square distance weights is usually larger than for the models with equal weights. It indicates that the most important objects for classification of a test object are the nearest neighbors but a number of farther objects can provide useful information too. The information from farther objects should only be considered less important than information from the nearest objects.

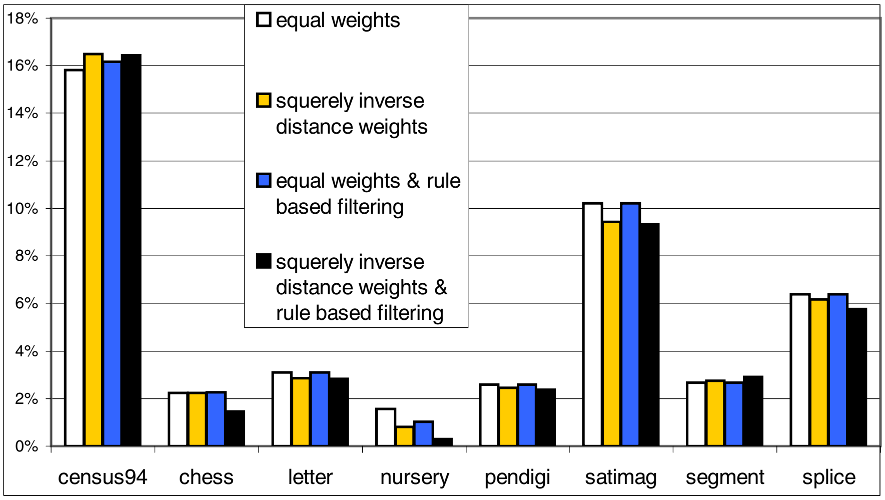


Fig. 27. The average classification error of the k-nn with the optimal k estimated from the training set for the joint city-block and VDM metric with the attribute weighting method optimizing distance

Figure 27 presents the average classification error of the k nearest neighbor method for the weighted joint city-block and VDM metric. The table below presents the best voting model for this metric and the confidence level of the differences between the best voting model and the others (see Subsection 2.7) for particular data sets:

The data set	The winning voting model	The confidence level
census94	equal weights	99.5%
chess	sqr. inv. distance weights & rule based filtering	99.5%
letter	sqr. inv. distance weights & rule based filtering	90% (from sqr. inv. dist.) 99.5% (from the two remaining)
nursery	sqr. inv. distance weights & rule based filtering	99.5%
pendigits	sqr. inv. distance weights & rule based filtering	¡90% (from sqr. inv. dist.) 90% (from the two remaining)
satimage	sqr. inv. distance weights & rule based filtering	90% (from sqr. inv. dist.) 99.5% (from the two remaining)
segment	equal weights both with and without rule based filtering	¡90% (from sqr. inv. dist.) 95% (from sqr. inv. dist. & rule based filter.)
splice	sqr. inv. distance weights & rule based filtering	90% (from sqr. inv. dist.) 99.5% (from the two remaining)

Figure 27 and the above table indicate that the voting model combining the inverse square distance weights with rule based filtering is the best: it gives the smallest error for six of the eight data sets: *chess, letter, nursery, pendigits, satimage* and *splice.* The most noticeable reduction in error is obtained for the two data sets: in case of the data set *nursery* the combined voting model gives the 0.3% error in comparison to the 1.57% error of the pure majority voting model with equal weights and in case of the data set *chess* the combined model gives the 1.46% error in comparison to the 2.24% error of the majority model.

While comparing the second and the third column in Figure 27 for each of the six data sets where the combined model is the best the model with the inverse square distance weights alone provides always a smaller error than the model with the equal weights and rule based filtering. This observation indicates that the inverse square distance weights are a more important component for the accuracy of the combined voting model than the rule based filtering. It is also confirmed by the fact that the difference between the combined voting model and the two models with equal weights (with and without rule based filtering) has almost always the maximum confidence level whereas the difference between the inverse square distance weights with and without rule based filtering has often a low confidence level. However, in case of the data sets *nursery* and *chess,* where the reductions in error by the combined voting model are largest, both components of this model contribute significantly to such good results.

Now, we consider the second metric: the weighted joint DBVDM and VDM metric. Since for data with nominal attributes this metric is equivalent to the weighted joint city-block and VDM metric we present the result only for data that contain numerical attributes. The table below presents the average value of the estimation of the optimal k for particular voting models with the weighted joint DBVDM and VDM metric:

Data set	equal weights	sqr. inv. dist. weights	equal weights & rule based filter.	sqr. inv. dist. weights & rule based filter.
census94	40.2	168.4	128.2	183
letter	1	4.4	1	6
pendigits	1	3.8	1	5.2
satimage	3.6	4	3.6	3.8
segment	1	1	1	4.8

We can make the same observation as in case of the weighted joint city-block and VDM metric: the voting models with the inverse square distance weights make use of the distance-dependent weights and they use more objects to vote than the models with equal weights.

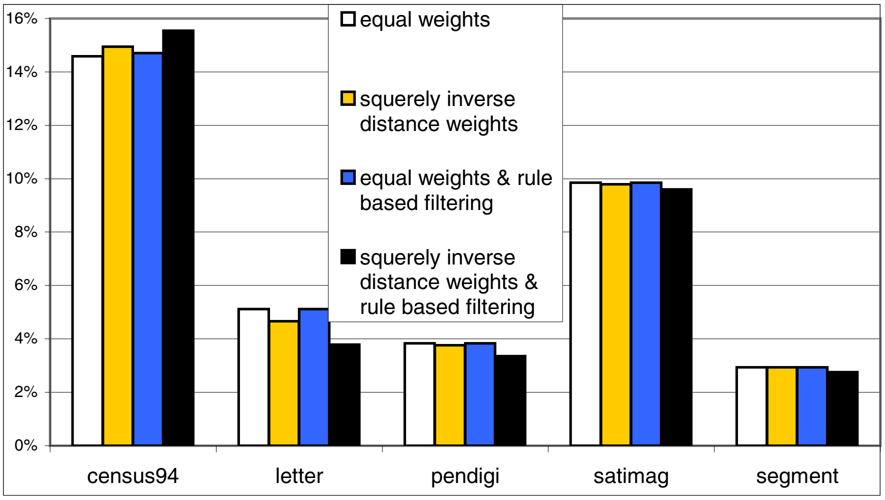


Fig. 28. The average classification error of the k-nn with the optimal k estimated from the training set for the joint DBVDM and VDM metric with the attribute weighting method optimizing distance

Figure 28 presents the average classification error of the k nearest neighbor method for the weighted joint DBVDM and VDM metric. The table below presents the best voting model for this metric and the confidence level of the difference between the best voting model and the others, for particular data sets:

The data set	The winning voting model	The confidence level
census94	equal weights	97.5% (from eq. weights (with rule based filter.) 99.5% (from the two remaining)
letter	inv. sqr. distance weights & rule based filtering	99.5%
pendigits	sqr. inv. distance weights & rule based filtering	99.5%
satimage	sqr. inv. distance weights & rule based filtering	95% (from sqr. inv. dist.) 99.5% (from the two remaining)
segment	sqr. inv. distance weights & rule based filtering	90%

As in case of the weighted joint city-block and VDM metric the results indicate that the voting model combining the inverse square distance weights with rule based filtering is the best: it gives the smallest error for all the data sets except for *census94*. The results are usually worse than for the weighted joint city-block and VDM metric. An interesting observation is that each of the two components of the combined metric: the inverse square distance weights and rule based filtering alone gives very small improvement (compare the second and the third column for each data set in Figure 28) and only the combination of these two components gives more noticeable reduction of the classification error.

The final conclusion from the presented results is that the voting model combining the inverse square distance weights with rule based filtering gives generally the best classification accuracy. It indicates that the significance of the information for a test object provided by the nearest neighbors correlates with the distance of the nearest neighbors to the test object and it is helpful to use this correlation. Distance measure and grouping of objects by rules are the two different sorts of similarity models and the application of rule based filtering to the nearest neighbors is a sort of combination of these two models. The neighbors selected for voting in such a combined method are similar to a test object according to both models, which gives more certainty that these neighbors are appropriate for decision making.

The above general conclusion does not fit to the results for the data set *census94*. This is related with the specificity of this data set. The estimated value of the optimal k for the data set *census94* is always much larger than for the other data sets. In case of such a large neighborhood the models with the inverse square distance weights are not enough accurate to improve the classification results and more accurate voting model should be searched.

5.6 *K* Nearest Neighbors with Local Metric Induction

All the variants of the k nearest neighbors method presented in the dissertation up to now and all other machine learning methods based on inductive concept learning: rule based systems, decision trees, neural networks, bayesian networks

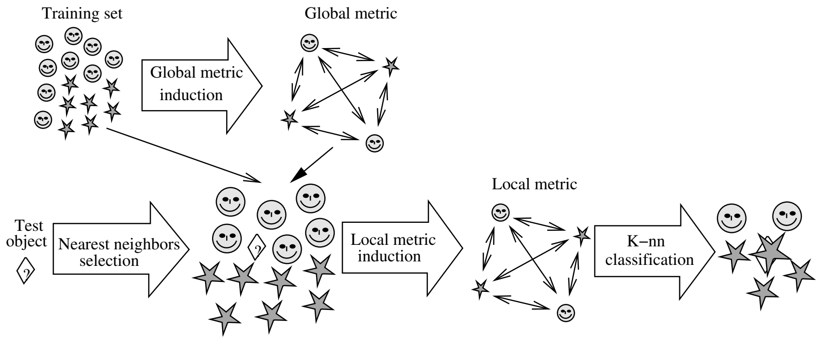


Fig. 29. *K*-nn classification with local metrics

and rough sets [59, 61, 63] induce a mathematical model from training data and apply this model to reasoning about test objects. The induced model of data remains invariant for different test objects. For many real-life data it is not possible to induce relevant global models. This fact has been recently observed by researches from different areas like data mining, statistics, multiagent systems [17, 75, 79]. The main reason is that phenomena described by real-life data are often too complex and we do not have sufficient knowledge to induce global models or a parameterized class of such models together with feasible searching methods for the relevant global model in such a class. Developing methods for dealing with such real-life data is a challenge.

In this subsection we propose a step toward developing of such methods. We propose a classification model that is composed of two steps. For a given test object x, first, a local model dependent on x is induced, and next, this model is used to classify x.

To apply this idea we extend the classical k-nn classification model described in Subsection 2.5. The classical k-nn induces a global metric ρ from the training set U_{trn}, and next, for each test object x it uses this induced metric ρ to find the k nearest neighbors of x and it computes a decision from the decisions of these k neighbors. We propose a new algorithm extending the classical k-nn with one additional intermediate step (see Figure 29). First, it induces a global metric ρ like in the classical k-nn method but this global metric ρ is used only in preliminary elimination of objects not relevant for classifying x. For each test object x the extended algorithm selects a neighborhood of x according to the global metric ρ and it induces a local metric ρ^x based only on the selected neighborhood. Local metric induction is a step to build a model that depends locally on the properties of the test object x. The final k nearest neighbors that are used to make a decision for the test object x are selected according to the locally induced metric.

A local approach to the k-nn method has been already considered in the literature. However, all the methods described in the literature apply only to data with numerical attributes and they assume always a specific metric to be defined.

Friedman proposed a method that combines the k-nn method with recursive partitioning used in decision trees [32]. For each test object the method starts with the whole training set and it constructs a sequence of partitions. Each partition eliminates a number of training objects. In this way after the last partition a small set of k objects remains to be used for classification. To make a single partition the algorithm selects the partition with the greatest decision discernibility.

The algorithm proposed by Hastie and Tibshirani [44] starts with the Euclidean metric and for each test object it iteratively changes the weights of attributes. At each iteration it selects a neighborhood of a test object and it applies local discriminant analysis to shrink the distance in the direction parallel to the boundary between decision classes. Finally, it selects the k nearest neighbors according to the locally transformed metric.

Domeniconi and Gunopulos use a similar idea but they use support vector machines instead of local discriminant analysis to determine class boundaries and to shrink the distance [24]. Support vectors can be computed during the learning phase what makes this approach much more efficient in comparison to local discriminant analysis.

As opposed to the above three methods our method proposed in this subsection is general: it assumes only that a procedure for metric induction from a set of objects is provided.

Algorithm 11. The k nearest neighbors algorithm $decision_{local-knn}(x)$ with local metric induction

```
$\rho$ - the global metric induced once
    from the whole training set $U_{trn}$
$l$ - the size of the neighborhood
    used for local metric induction
$k_{opt}$ - the optimal value of $k$ estimated
      from the training set $U_{trn}$ ($k_{opt} \leq l$)

$NN(x,l) :=$ the set of $l$ nearest neighbors of $x$ from $U_{trn}$
          according to the global metric $\rho$
$\rho^x :=$ the local metric induced from the neighborhood $NN(x,l)$
$NN_{local}(x,k_{opt}) :=$ the set of $k_{opt}$ nearest neighbors of $x$
            from $NN(x,l)$ according to the local metric $\rho^x$
return $\arg\max_{d_j \in V_{dec}} |\{y \in NN_{local}(x,k_{opt}) : \ dec(y) = d_j\}|$
```

In the learning phase our extended method induces a global metric ρ and estimates the optimal value k_{opt} of nearest neighbors to be used for classification. This phase is analogous to the classical k-nn.

Algorithm 11 presents the classification of a single query object x by the method extended with local metric induction. First, the algorithm selects the l nearest neighbors $NN(x,l)$ of x from the training set U_{trn} according to the global metric ρ. Next, it induces a local metric ρ^x using only the selected neighborhood $NN(x,l)$. After that the algorithm selects the set $NN_{local}(x,k_{opt})$ of the nearest neighbors of x from the previously selected neighborhood $NN(x,l)$ according to this local metric ρ^x . Then, the selected set $NN_{local}(x, k_{opt})$ is used to compute the decision $decision_{local-knn}(x)$ that is returned as the final result for the query object x.

Both for the global and for the local metric definition the algorithm can use any metric induction procedure. Moreover, different metrics can be used in the global and in the local step.

The neighborhood size l is the parameter of the extended method. To improve classification accuracy this value should be large, usually at least of an order of several hundred objects. To accelerate the selection of a large number of nearest neighbors from a training set we use the indexing tree with the iterative 3-means splitting procedure and the combined search pruning criteria described in Section 4.

The optimal value k_{opt} is estimated from a training set within the range $1 \leq k \leq l$ with the use of the same efficient procedure as in case of the classical k-nn presented in Algorithm 7 in Subsection 5.1. However, the estimation process uses Algorithm 11 as the classification procedure instead of the classical k-nn classification procedure from Equation 4. This is the only difference between the learning phases of the classical and the extended method.

In Algorithm 11 we use the most popular majority voting model with equal weights. However, as in the classical k-nn method any voting model can be used in the method with local metric induction.

The classical k-nn is a lazy method: it induces a global metric and it performs the rest of computation at the moment of classification. The algorithm proposed in this subsection extends this idea: it repeats the metric induction at the moment of classification. The proposed extension allows us to use the local properties of data topology in the neighborhood of a test object and to adjust the metric definition to these local properties.

5.7 Comparison of *k*-nn with Global and with Local Metric

In this subsection we compare the performance of the k nearest neighbors method with the local metric induction described in the previous subsection and the performance of the classical k-nn method. We compare the classical k-nn and the extended k-nn with the three different values of the neighborhood size l: 100, 200 and 500.

To compare the methods we tested the 10 benchmark data sets presented in Table 1. As in all the previous experiments described in the dissertation each data set was partitioned into a training and a test set as described in Subsection 2.7. Then training and testing for each data set and for each classification method was performed 5 times for the same partition of the data set and the average classification error was calculated for comparison. For the classical k-nn method the optimal value k_{opt} was estimated in the range $1 \leq k_{opt} \leq 200$ and for the extended method for each of the values of l: 100, 200 and 500 the optimal value k_{opt} was estimated in the range $1 \leq k_{opt} \leq l$.

The two most effective global metrics were tested: the joint city-block and VDM metric (see Subsection 3.1) with the attribute weighting method optimiz-

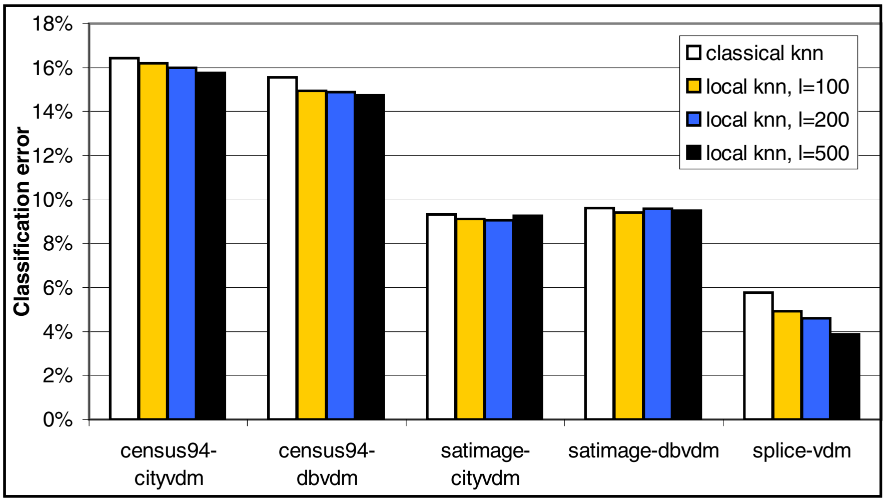


Fig. 30. The average classification error of the classical and the extended k-nn with the two metrics: The weighted joint city-block and VDM metric and the weighted joint DBVDM and VDM metric as the global metric, and with the weighted joint city-block and VDM metric as the local metric, obtained for the three different neighborhood sizes: 100, 200 and 500

ing distance (see Subsection 3.4) and the joint DBVDM and VDM metric (see Subsection 3.2) with the same attribute weighting method. Using the weighted joint DBVDM and VDM metric as the local metric makes the k nearest neighbors impractical: the performance time of k-nn with this metric becomes too long. Therefore the weighted joint city-block and VDM metric was always used as the local metric. Since the voting model combining the inverse square distance weights with rule based filtering provides generally the best classification accuracy (see Subsection 5.5) we apply this voting model in the experiment.

For the seven of the 10 tested data sets: *chess, letter, mushroom, nursery, pendigits, segment* and *shuttle* the classical k-nn method with the combined voting model obtained the classification accuracy over 97% (see Subsection 5.5). Such a good accuracy is hard to improve and for these seven data sets the k nearest neighbors with local metric induction does not provide better results or the improvement in accuracy is insignificant. Therefore we focus on the three most difficult data sets: *census94* (16.44% error by the weighted joint city-block and VDM metric and 15.54% error by the weighted joint DBVDM and VDM metric), *satimage* (9.33% error by the weighted city-block metric and 9.6% error by the weighted DBVDM metric) and *splice* (5.77% error by the weighted VDM metric).

Figure 30 presents the average classification errors of the classical and the extended k-nn method with the two types of the global metric. For the three presented data sets the extended method provides always better classification accuracy than the classical method. The table below presents the confidence level of the difference between each of the extended method and the classical k-nn (see Subsection 2.7) for each data set:

Data set Global metric	local k-nn (l=100) vs. classical k-nn	local k-nn (l=200) vs. classical k-nn	local k-nn (l=500) vs. classical k-nn
census94 city-block & VDM	99.5%	99.5%	99.5%
census94 DBVDM & VDM	99.5%	99.5%	99.5%
satimage city-block	95%	99%	¡90%
satimage DBVDM	¡90%	¡90%	¡90%
splice VDM	99.5%	99.5%	99.5%

The best improvement was obtained for the data set *splice*. The difference between the extended and the classical k-nn has always the maximum confidence level and in the best case of $l = 500$ the extended method reduced the classification error from 5.77% to 3.86%. This result is particularly noteworthy: the author has never met such a good result in the literature for this data set. For the data set *census94* the difference between the extended and the classical k-nn has always the maximum confidence level too. For this data set the improvement by the extended method is not so large but it is still noticeable: in the best case of $l = 500$ for the weighted joint DBVDM and VDM metric the classification error was reduced from 15.54% to 14.74%. The least effect of applying local metric induction one can observed for the data set *satimage*: only in one case the statistical significance of the difference between the extended and the classical method is trustworthy: in case of $l = 200$ for the weighted city-block metric the classification error was reduced from 9.33% to 9.07%.

An interesting observation is that the largest improvement was obtained for data only with nominal attributes and the smallest improvement was obtained for data only with numerical attributes. It correlates with the fact that the general properties of the domain of values of nominal attributes are poor and the methods for data with nominal attributes learn mainly from the information encoded in data. Hence, a metric induced globally from the whole training set and a metric induced locally from a neighborhood of a test object can differ significantly, the local metric can adapt strongly to local properties of the neighborhood and thus the possibility of improving accuracy by the local metric is large. In case of numerical attributes there are a structure of linear order and a distance measure defined in the set of values. In many cases this structure corresponds well with the properties of objects important for the decision attribute. The weighted city-block metric is consistent with this structure and it is often enough to apply this metric in order to obtain almost optimal classification accuracy. Therefore improving a global metric by local metric induction in case of data with numerical attributes is much more difficult than in case of data with nominal attributes.

An interesting issue is the dependence between the classification accuracy and the neighborhood size l used to induce a local metric. For the two data sets: *census94* and *splice* where the improvement by local metric induction is significant the best results was obtained for the maximum tested neighborhood size $l = 500$. It indicates that an important factor for the quality of a local metric is the representativeness of the sample used for metric induction and it is important to balance between the locality and the representativeness of the neighborhood used to induce a local metric.

5.8 Summary

In this section we have introduced two new classification models based on the k nearest neighbors:

– k nearest neighbors method combined with rule based filtering of the nearest neighbors,
– k nearest neighbors method based on a locally induced metric.

In the beginning, we have presented the algorithm estimating the optimal value of k from training data. The algorithm allows us to set automatically an appropriate value of k.

Then we have considered different voting models known from the literature. The most popular is the majority voting model where all the k nearest neighbors are weighted with equal weights. The distance based voting model replaces equal weights by the inverse square distance weights. The classification accuracy of the distance based voting model is better than the majority voting model what reflects the fact that the significance of the information for a test object provided by the nearest neighbors correlates with the distance of the nearest neighbors to the test object and it is helpful to use this correlation.

The first new model introduced in this section adds rule based filtering of the k nearest neighbors to the classical k-nn method. As the origin we took Bazan's lazy algorithm simulating effectively the classification model based on all minimal consistent rules for data with nominal attributes and we generalized the equality based model of minimal consistent rules to the metric based model. Next, we adapted Bazan's algorithm to the metric based model of minimal consistent rules, and finally, we attached this generalized rule based classification model to the k nearest neighbors in the form of rule based filtering of the k nearest neighbors. An important property of the proposed combination is that the addition of rule based filtering does not change essentially the performance time of the k nearest neighbors method. The experimental results show that the application of rule based filtering improves the classification accuracy especially when combined with the voting model with the inverse square distance weights. It indicates that rule based filtering improves selection of objects for reasoning about a test object.

The estimation of the optimal value of k and all the described voting models are available in the k nearest neighbors classifier in the system RSES [8, 73].

The classifier makes it possible to choose between the model with equal weights and the model with the inverse square distance weights, and optionally, it allows us to apply rule based filtering to the k nearest neighbors for each test object.

As the second method we proposed a new classification model that is an extension of the classical k-nn classification algorithm. The extended method induces a different metric for each test object using local information in the neighborhood of an object. The k-nn model with a local metric corresponds to the idea of transductive reasoning [79]. The transductive approach assumes that a classification model should depend on the objects to be classified and it should be adapted according to the properties of these objects. The presented extension of the k-nn algorithm implements transduction: the local metric induction adapts the metric definition to the local topology of data in the neighborhood of an object to be classified.

The experimental results show that the local approach is particularly useful in the case of hard problems. If the classification error of the methods based on global models remains large a significant improvement can be obtained with the local approach.

An important problem related to the k nearest neighbors method with local metric induction is that the local metric induction for each test object is a time-consuming step. As a future work we consider the extension of data representation in such a way that the algorithm can use the same local metric for similar test objects.

6 Conclusions

In the dissertation we have developed different classification models based on the k-nn method and we have evaluated them against real-life data sets.

Among the k-nn voting models based on the global metric the most accurate model can be obtained by the method combining the inverse square distance weights with the nearest neighbors filtering performed by means of the set of minimal consistent rules. The assignment of the inverse square distance weights to the nearest neighbors votes reflects the fact that the more similar a test object is to a training object, the more significant is for the test object the information provided by the training object. The rule-based filtering method introduced in the dissertation makes it possible to construct an alternative model that combined with the k-nn method enables verification of objects recognized by the k-nn as similar and the rejection of the objects that are not confirmed to be similar by the rule based model. The proposed rule-based extension is independent of the metric and does not increase the performance time of the classical k-nn method. Therefore it can be applied whenever the classical k-nn is applicable.

We compared different metrics in the k-nn classification model. In general, the best metrics are the normalized city-block metric for numerical attributes and the Value Difference Metric for nominal attributes, both combined with attribute weighting. For nominal attributes there is no mathematical structure in the domain of values, therefore the Value Difference Metric uses the information encoded in the training data to measure similarity between nominal values. Domains with numerical values have the structure of linear order and a distance measure consistent with this linear order. These properties reflect usually the natural relationship among the values of the numerical attribute and this information is often sufficient to define an accurate similarity measure for the values of a numerical attribute. However, there are decision problems where the natural metric on numerical values does not reflect directly the differences between decision values. For such data the correlation between the attribute values and the decision encoded in training objects is the information more useful than the general structure of numerical values. Hence, solutions analogous to the Value Difference Metric for nominal attributes are more accurate. We have analyzed three numerical metrics of this type: IVDM, WVDM, and DBVDM. They estimate decision probabilities for particular numerical values in the training set and use these estimations to define distance between values. In DBVDM the sample for decision probability estimation is chosen more carefully than in IVDM and WVDM and it gives the most accurate classification among these three metrics.

There are hard classification problems where the relationship between attributes and the decision is complex and it is impossible to induce a sufficiently accurate global model from data. For such data the method with local model induction is a better solution. The algorithm yields a separate local decision model for each test object. This approach allows us to adapt each local model to the properties of a test object and thus to obtain a more accurate classification than in the case of the global approach.

To apply metric-based classification models to large databases a method that would speed up the nearest neighbors search is indispensable. The extension of the indexing and searching methods described in literature, i.e., the iterative splitting based tree with three search pruning criteria, as proposed in the dissertation, allows us to use the k-nn method to data sets with several hundred thousand objects.

The results presented in the dissertation do not exhaust all the aspects of case-based reasoning from data. The following extensions can be considered.

Experiments with different metrics for numerical data have proved that the city-block metric is more accurate than metrics that do not preserve consistency with the natural linear order of the real numbers. However, such a metric uses the natural metric of real numbers and the training set is used marginally to modify this natural metric. An interesting issue is to construct and investigate metrics preserving the natural order of numerical values and to use training data to differentiate the value of the distance in dependence on the range of values to be compared.

A more general problem related to metric construction is that the induction of an accurate metric only from data without additional knowledge is impossible for more complex decision problems. Therefore the development of methods for inducing similarity models based on interaction with a human expert acquires particular importance.

Another possible continuation is related to the induction of local classification models. The method presented in the dissertation induces a separate model for each test object. Such a solution is computationally much more expensive than methods based on the global model. Another possible solution is to use the common sense assumption that a local model can be relevant for a fragment of a space of objects. Such an approach has been already used in data mining [49] where transactions are partitioned into groups and a specific optimization function is defined for each group. By analogy, one can partition training objects into groups of similar objects and construct one local classification model for each group. This approach integrated with an indexing structure could be comparable to the global k-nn method in terms of efficiency.

Acknowledgments

The paper is the full version of my phd dissertation supervised by Andrzej Skowron, approved in May 2005 by Warsaw University, Faculty of Mathematics, Informatics and Mechanics.

I wish to express my gratitude to my supervisor Professor Andrzej Skowron for leading me towards important research issues and for his guidance in my research work. His attitude to me has been invariably kind and gentle, and Professor has always enabled me to pursue my own research endeavors. I am also indebted to him for making such great efforts to thoroughly revise this dissertation. I am happy that I have been given the opportunity to work with Professor Andrzej Skowron.

I thank my wife, Ewa. Her sincere love, support and forbearance helped me very much in writing this dissertation.

I am also grateful to many other people who contributed to my PhD dissertation. Dr Marcin Szczuka has supported me with hardware and software necessary for performing experiments. The pleasant and fruitful cooperation with Grzegorz Góra helped me determine my research objectives. Long discussions with dr Jan Bazan, Rafal Latkowski, and Michal Mikolajczyk, on the design and implementation of the programming library *rseslib* allowed me to make it useful for other researchers. Professor James Peters, dr Wiktor Bartol and Joanna Kuźnicka revised thoroughly the language of the dissertation.

Financial support for this research has been provided by the grants 8 T11C 009 19, 8 T11C 025 19, 4 T11C 040 24 and 3 T11C 002 26 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. Ch. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behaviour of distance metrics in high dimensional space. In *Proceedings of the Eighth International Conference on Database Theory*, pages 420–434, London, UK, 2001.
2. D. W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.

3. D. W. Aha. The omnipresence of case-based reasoning in science and applications. *Knowledge-Based Systems*, 11(5-6):261–273, 1998.
4. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
5. K. Ajdukiewicz. *Logika Pragmatyczna.* PWN, Warszawa, 1974.
6. J. G. Bazan. Discovery of decision rules by matching new objects against data tables. In *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing*, volume 1424 of *Lectures Notes in Artificial Intelligence*, pages 521–528, Warsaw, Poland, 1998. Springer-Verlag.
7. J. G. Bazan and M. Szczuka. RSES and RSESlib - a collection of tools for rough set computations. In *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing*, volume 2005 of *Lectures Notes in Artificial Intelligence*, pages 106–113, Banff, Canada, 2000. Springer-Verlag.
8. J. G. Bazan, M. Szczuka, A. G. Wojna, and M. Wojnarski. On the evolution of Rough Set Exploration System. In *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing*, volume 3066 of *Lectures Notes in Artificial Intelligence*, pages 592–601, Uppsala, Sweden, 2004. Springer-Verlag.
9. N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The $R^{\star}$-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, 1990.
10. R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.
11. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
12. S. Berchtold, D. Keim, and H. P. Kriegel. The X-tree: an index structure for high dimensional data. In *Proceedings of the Twenty Second International Conference on Very Large Databases*, pages 28–39, 1996.
13. K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the Seventh International Conference on Database Theory*, pages 217–235, Jerusalem, Israel, 1999.
14. Y. Biberman. A context similarity measure. In *Proceedings of the Ninth European Conference on Machine Learning*, pages 49–63, Catania, Italy, 1994.
15. C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, England, 1996.
16. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, CA, 1998.
17. L. Breiman. Statistical modeling - the two cultures. *Statistical Science*, 16(3):199–231, 2001.
18. S. Brin. Near neighbor search in large metric spaces. In *Proceedings of the Twenty First International Conference on Very Large Databases*, pages 574–584, 1995.
19. E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. Technical Report TR/DCC-99-3, Department of Computer Science, University of Chile, 1999.
20. P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proceedings of the Twenty Third International Conference on Very Large Databases*, pages 426–435, 1997.
21. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–284, 1989.

22. S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
23. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
24. C. Domeniconi and D. Gunopulos. Efficient local flexible nearest neighbor classification. In *Proceedings of the Second SIAM International Conference on Data Mining*, 2002.
25. P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.
26. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, NY, 1973.
27. S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6:325–327, 1976.
28. R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.
29. R. Finkel and J. Bentley. Quad-trees: a data structure for retrieval and composite keys. *ACTA Informatica*, 4(1):1–9, 1974.
30. R. A. Fisher. Applications of "student"s' distribution. *Metron*, 5:3–17, 1925.
31. E. Fix and J. L. Hodges. Discriminatory analysis, non-parametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation and Medicine, Randolph Air Field, 1951.
32. J. Friedman. Flexible metric nearest neighbor classification. Technical Report 113, Department of Statistics, Stanford University, CA, 1994.
33. J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning.* Springer, New York, NY, 2001.
34. J. H. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 717–724, Cambridge, 1996.
35. K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24(7):750–753, 1975.
36. V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
37. A. R. Golding and P. S. Rosenbloom. Improving accuracy by combining rule-based and case-based reasoning. *Artificial Intelligence*, 87(1-2):215–254, 1996.
38. G. Góra and A. G. Wojna. Local attribute value grouping for lazy rule induction. In *Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing*, volume 2475 of *Lectures Notes in Artificial Intelligence*, pages 405–412, Penn State Great Valley, PA, 2002. Springer-Verlag.
39. G. Góra and A. G. Wojna. RIONA: a classifier combining rule induction and k-nn method with automated selection of optimal neighbourhood. In *Proceedings of the Thirteenth European Conference on Machine Learning*, volume 2430 of *Lectures Notes in Artificial Intelligence*, pages 111–123, Helsinki, Finland, 2002. Springer-Verlag.
40. G. Góra and A. G. Wojna. RIONA: a new classification system combining rule induction and instance-based learning. *Fundamenta Informaticae*, 51(4):369–390, 2002.
41. "Student" (W. S. Gosset). The probable error of a mean. *Biometrika*, 6:1–25, 1908.
42. J. W. Grzymala-Busse. LERS - a system for learning from examples based on rough sets. In R. Slowinski, editor, *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, pages 3–18. Kluwer Academic Publishers, Dordrecht, Boston, London, 1992.

43. A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, 1984.
44. T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.
45. F. V. Jensen. *An Introduction to Bayesian Networks.* Springer Verlag, New York, 1996.
46. I. Kalantari and G. McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9(5):631–634, 1983.
47. N. Katayama and S. Satoh. The SR-tree: an index structure for high dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 369–380, Tucson, Arizona, 1997.
48. K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1992. Morgan Kaufmann.
49. J. Kleinberg, Ch. Papadimitriou, and P. Raghavan. Segmentation problems. *Journal of the ACM*, 51(2):263–280, 2004.
50. W. Klösgen and J. M. Żytkow, editors. *Handbook of Data Mining and Knowledge Discovery.* Oxford University Press, Inc., New York, NY, USA, 2002.
51. I. Kononenko. Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the Seventh European Conference on Machine Learning*, volume 784 of *Lectures Notes in Artificial Intelligence*, pages 171–182, Catania, Italy, 1994. Springer-Verlag.
52. D. B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons and Future Directions.* AAAI Press/MIT Press, 1996.
53. J. Li, G. Dong, K. Ramamohanarao, and L. Wong. DeEPs: a new instance-based discovery and classification system. *Machine Learning*, 2003. to appear.
54. J. Li, K. Ramamohanarao, and G. Dong. Combining the strength of pattern frequency and distance for classification. In *Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 455–466, Hong Kong, 2001.
55. K. I. Lin, H. V. Jagadish, and C. Faloustos. The TV-tree: an index structure for high dimensional data. *VLDB Journal*, 3(4):517–542, 1994.
56. D. Lowe. Similarity metric learning for a variable kernel classifier. *Neural Computation*, 7:72–85, 1995.
57. D. R. Luce and H. Raiffa. *Games and Decisions.* Wiley, New York, 1957.
58. J. E. S. Macleod, A. Luk, and D. M. Titterington. A re-examination of the distance-weighted k-nearest-neighbor classification rule. *IEEE Transactions on Systems, Man and Cybernetics*, 17(4):689–696, 1987.
59. R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.
60. R. S. Michalski, I. Mozetic, J. Hong, and H. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045, 1986.
61. T. M. Mitchell. *Machine Learning.* McGraw-Hill, Portland, 1997.
62. J. Nievergelt, H. Hinterberger, and K. Sevcik. The grid file: an adaptable symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, 1984.
63. Z. Pawlak. *Rough Sets - Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers, Dordrecht, 1991.

64. L. Polkowski and A. Skowron. Synthesis of decision systems from data tables. In T. Y. Lin and N. Cercone, editors, *Rough Sets and Data Mining: Analysis of Imprecise Data*, pages 259–299. Kluwer Academic Publishers, Dordrecht, 1997.
65. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
66. J. Robinson. The K-D-B-tree: a search structure for large multi-dimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, pages 10–18, New York, 1981.
67. A. Rosenblueth, N. Wiener, and J. Bigelow. Behavior, purpose, and teleology. *Philosophy of Science*, 10:18–24, 1943.
68. S. J. Russell. *Use of Knowledge in Analogy and Induction.* Morgan Kaufmann, 1989.
69. S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 2:229–246, 1991.
70. S. M. Savaresi and D. L. Boley. On the performance of bisecting K-means and PDDP. In *Proceedings of the First SIAM International Conference on Data Mining*, pages 1–14, Chicago, USA, 2001.
71. T. Sellis, N. Roussopoulos, and C. Faloustos. The R+-tree: a dynamic index for multi-dimensional objects. In *Proceedings of the Thirteenth International Conference on Very Large Databases*, pages 574–584, 1987.
72. R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237:1317–1323, 1987.
73. A. Skowron et al. Rough set exploration system. http://logic.mimuw.edu.pl/˜rses, Institute of Mathematics, Warsaw University, Poland.
74. A. Skowron and C. Rauszer. The discernibility matrices and functions in information systems. In R. Slowinski, editor, *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, pages 331–362. Kluwer Academic Publishers, Dordrecht, 1992.
75. A. Skowron and J. Stepaniuk. Information granules and rough-neural computing. In *Rough-Neural Computing: Techniques for Computing with Words*, Cognitive Technologies, pages 43–84. Springer-Verlag, Heidelberg, Germany, 2003.
76. A. Skowron and A. G. Wojna. K nearest neighbors classification with local induction of the simple value difference metric. In *Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing*, volume 3066 of *Lectures Notes in Artificial Intelligence*, pages 229–234, Uppsala, Sweden, 2004. Springer-Verlag.
77. C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.
78. J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.
79. V. Vapnik. *Statistical Learning Theory.* Wiley, Chichester, GB, 1998.
80. M. Veloso. *Planning and Learning by Analogical Reasoning.* Springer, 1994.
81. J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, New Jersey, 1944.
82. J. Ward, Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.
83. R. Weber, H. J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the Twenty Fourth International Conference on Very Large Databases*, pages 194–205, 1998.

84. D. Wettschereck. *A Study of Distance-Based Machine Learning Algorithms.* PhD thesis, Oregon State University, 1994.
85. D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.
86. D. A. White and R. Jain. Similarity indexing with the SS-tree. In *Proceedings of the Twelve International Conference on Data Engineering*, pages 516–523, New Orleans, USA, 1996.
87. N. Wiener. *Cybernetics.* Wiley, New York, 1948.
88. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.
89. D. R. Wilson and T. R. Martinez. An integrated instance-based learning algorithm. *Computational Intelligence*, 16(1):1–28, 2000.
90. A. G. Wojna. Adaptacyjne definiowanie funkcji boolowskich z przykladow. Master's thesis, Warsaw University, 2000.
91. A. G. Wojna. Center-based indexing for nearest neighbors search. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 681–684, Melbourne, Florida, USA, 2003. IEEE Computer Society Press.
92. A. G. Wojna. Center-based indexing in vector and metric spaces. *Fundamenta Informaticae*, 56(3):285–310, 2003.
93. D. Wolpert. Constructing a generalizer superior to NETtalk via meithematical theory of generalization. *Neural Networks*, 3:445–452, 1989.
94. J. Wróblewski. Covering with reducts - a fast algorithm for rule generation. In *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing*, volume 1424 of *Lectures Notes in Artificial Intelligence*, pages 402–407, Warsaw, Poland, 1998. Springer-Verlag.
95. P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 311–321, Austin, Texas, 1993.
96. J. Zavrel. An empirical re-examination of weighted voting for k-nn. In *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning*, pages 139–148, Tilburg, The Netherlands, 1997.

Appendix. List of Symbols Used in the Dissertation

$|\ldots|$ — size of a set
$\|\ldots\|_p$ — norm of a vector in the space l_p
μ_i — mean of a numerical attribute a_i in a training set U_{trn}
ρ — distance function $\mathbb{X}^2 \to \mathbb{R}$
ρ_i — distance function $\mathbb{R}^2 \to \mathbb{R}$ defined for the values of an attribute a_i
σ_i — standard deviation of a numerical attribute a_i in a training set U_{trn}
$\sigma(X)$ — standard deviation of a continuous variable X
a_i — single attribute
A — set of attributes
c_j — center of the j-th cluster Cl_j in a node splitting procedure
Cl_j — j-th cluster of data objects in a node splitting procedure
dec — decision function $\mathbb{X} \to V_{dec}$ to be learnt by classifiers
df — degree of freedom in the Student's t-test

d_j — single decision value

$E(X)$ — expected value of a continuous variable X

I_p — p-th interval at discretization of a numerical attribute for the IVDM metric

$\overline{I(v)}$ — index of the upper neighboring interval of a value v for the IVDM metric

$\underline{I(v)}$ — index of the lower neighboring interval of a value v for the IVDM metric

k — number of nearest neighbors in the k-nn classifier

k_{max} — upper limit of the range of values examined by the procedure estimating the optimal k

l — number of iterations in attribute weighting algorithms

max_i — maximum value of an attribute a_i in a training set U_{trn}

mid_p — midpoint of the interval I_p

min_i — minimum value of an attribute a_i in a training set U_{trn}

MR — global misclassification ratio

$MR(a_i)$ — misclassification ratio for an attribute a_i

$NN(x,k)$ — set of the k nearest neighbors of a data object x in a training set U_{trn}

$P(dec = d_j|a_i = v)$ — conditional decision probability given a value v of an attribute a_i

$P(dec = d_j|a_i \in I)$ — conditional decision probability given an interval of values I of an attribute a_i

$P_{DBVDM}(dec = d_j|a_i = v)$ — estimated conditional decision probability in the DBVDM metric

$P_{IVDM}(dec = d_j|a_i = v)$ — estimated conditional decision probability in the IVDM metric

$P_{VDM}(dec = d_j|a_i = v)$ — estimated conditional decision probability in the VDM metric

$P_{WVDM}(dec = d_j|a_i = v)$ — estimated conditional decision probability in the WVDM metric

$r_{local(x,y)}$ — local rule for a pair of a test object x and a training object $y \in U_{trn}$

$r_{gen-local(x,y)}$ — generalized local rule for a pair of a test object x and a training object $y \in U_{trn}$

R_x — number of generalized minimal consistent rules centered at x

$\mathbb{R}$ — set of real numbers

s — number of intervals at discretization of a numerical attribute for the IVDM metric

$support(r)$ — set of all objects in U_{trn} matching the rule r

t — value t in the Student's t-test

U_{trn} — training set

U_{tst} — test set

V_{dec} — set of decision values

V_i — domain of values of the attribute a_i

x_i — value of an attribute a_i in a data object $x \in \mathbb{X}$

$\mathbb{X}$ — space of data objects, domain of learning