Unsupervised and supervised data classification via nonsmooth and global optimization[1]

A. M. Bagirov, A. M. Rubinov, N.V. Soukhoroukova and J. Yearwood

School of Information Technology and Mathematical Sciences,
The University of Ballarat, Vic 3353, Australia.

Abstract

We examine various methods for data clustering and data classification that are based on the minimization of the so-called cluster function and its modifications. These functions are nonsmooth and nonconvex. We use Discrete Gradient methods for their local minimization. We consider also a combination of this method with the cutting angle method for global minimization. We present and discuss results of numerical experiments.

Key words: clustering, classification, cluster function, nonsmooth optimization, global optimization

1 Introduction

With the rapid increase in the availability of data for exploration and analysis it is important to develop techniques that efficiently perform data clustering and data classification. In the first case there is a need to get insight into the data and find out things about it in as objective a way as possible. This type of approach is usually classed as exploratory data analysis and includes clustering. The second concerns learning or identifying the extent to which the data conforms to known or hypothesized models.

Clustering or cluster analysis involves the identification of subsets of the data that are similar. The subset usually intuitively corresponds to points that are more similar to each other than they are to points from another cluster. Points in the same cluster have the same label. Clustering is carried out in an *unsupervised* way by trying to find subsets of points that are similar without having a predefined notion of the cluster. We can say that the identification of these clusters with labels is *data driven* rather than determined by a particular model or view of the data.

Classification involves the *supervised* assignment of data points to predefined and known classes. Here, there is a collection of classes with labels and the problem is to label a new observation or data point as belonging to one or more of the classes. Usually the known classes of examples constitute a *training set* and are used to learn a description of the classes. This can then be used to assign new examples to classes. The classes are determined by some a priori knowledge about the dataset.

[1]This research was supported by the Australian Research Council

1.1 Clustering

According to Jain [51], the clustering task usually involves the following steps:

1. representation of the data;
2. deciding on a similarity measure or distance metric that is most appropriate for the task in the domain;
3. performing the clustering;
4. cluster description;
5. evaluation.

Data representation is the task of deciding the number of features available, their nature and scale, the size of the dataset and the number of clusters or classes. *Feature selection* is the process of selecting a set of features that are the most effective subset to use with the clustering algorithm. Feature selection methods are used to identify the most informative features, to remove some uninformative or noisy features and reduce the dimension of the problem under consideration. *Feature extraction or feature combination* is the use of transformations of the feature set to produce a set of features that are able to be effectively used by the clustering algorithm.

In many problems of cluster analysis, there is little prior information available about the data, and as few assumptions about the data as possible can be made. It is under these restrictions that clustering methodology appropriate for the exploration of interrelationships among data points can be used to make assessments of their structures.

The notion of clustering is relatively *flexible* as the aim is to identify and reveal clusters or groups in an exploratory data analysis sense. An important concept is that of a cluster representative also called cluster profile, classification vector, cluster label, or centroid. It is simply an object that summarises and represents the objects in the cluster. It should be close to every object in the cluster in some average sense. The similarity of the objects to the objects is measured by a matching function or similarity function.

The notion of similarity is crucial in the definition of clustering. *Similarity* is usually measured by some dissimilarity measure such as a metric defined on the dataset. However there are other ways of approaching the definition of similarity. For example, Finnie and Sun [41] define a similarity relation as an equivalence relation. This would then imply that similarity is a transitive notion which is stronger than what we usually expect in the notion of similarity.

Most clustering algorithms use a number of empirically determined parameters such as:

- the number of clusters
- a minimum and maximum size of each cluster
- a threshold value on the matching function, below which an object will not be included in a cluster
- a control on overlap between clusters
- a objective function which is optimised

1.2 Clustering Techniques

In clustering, data or observations are usually represented as *feature vectors* and the individual scalar components of a feature vector $x = (x_1, \ldots x_n)$ are called features or attributes. The dimensionality of the data or the dataset is n.

In selecting or developing clustering techniques the method should exhibit some theoretical soundness. This may be assessed by certain criteria of adequacy. Some criteria that have been suggested by Jardine and Sibson [52] are:

1. the method produces a clustering which is unlikely to be altered drastically when additional objects are incorporated - *stability under growth*;
2. the method is stable in the sense that small perturbations in the description of the objects lead to small changes in the clustering;
3. the method is independent of the initial ordering of the objects.

1.2.1 Hierarchical Clustering

A hierarchical algorithm produces a *dendrogram* presenting a nested grouping of data and similarity levels at which the clusters change. Most hierarchical clustering algorithms are variants of the single link [85], the complete link [56] and minimum variance [88, 71] algorithms.

The single-link approach takes as the distance between two clusters, the *minimum* of the distances between all pairs of points in the two clusters (one from the first cluster, the other from the second). The complete-link algorithm takes the distance between clusters as the *maximum* of all pairwise distances between points in the two clusters. The complete-link algorithm produces compact clusters [8] whereas the single link algorithm tends to produce clusters that are elongated due to a chaining effect [72].

1.2.2 Partitional Algorithms

A partitional clustering algorithm produces a single partition of the data with no hierarchical structure. This means that the algorithm usually requires the number of clusters to be specified. They usually optimize a criterion function defined on the dataset. Most algorithms have traditionally been run multiple times with different starting states and the best configuration produced is the one used as the clustering. The most frequently used objective function is the squared error criterion

$$f_k(x) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \|a_i^{(j)} - x_j\|^2,$$

where k is the number of clusters, n_j is the number of records in the cluster j, $j = 1, \ldots, k$, $a_i^{(j)}$ is the i-th element of the cluster j, $i = 1, \ldots, n_j$ and x_j is the centroid of the j^{th} cluster: $x_j = (1/n_j) \sum_{i=1}^{n_j} a_i^{(j)}$. This has been found to work well with isolated and compact clusters. The k-means algorithm is the most commonly used algorithm using this criterion [66]

The k-means algorithm randomly chooses k cluster centres and iteratively reassigns data points to clusters based on the similarity between the pattern and the cluster centers until

there is no further reassignment or the squared error no longer decreases significantly. It is popular because it is easy to implement with time complexity $O(n)$. It suffers from being sensitive to the selection of the initial clustering partition or cluster centres. Several variants of the k-means algorithm have been reported in the literature [3], many of them focussing on the selection of a good initial partition.

1.2.3 k-Nearest Neighbour Clustering

In this approach, each unlabelled data point is assigned to the cluster of its k nearest labelled neighbours as long as the average distance to the k neighbours is below a threshold [60].

1.2.4 Connectionist techniques

Artificial neural networks (ANNs) have been used extensively for both classification and clustering [83]. The most common ANNs used for clustering include Kohonen's learning vector quantization (LVQ) and self organizing map (SOM) [58] and adaptive resonance theory (ART) models [33]. These networks have simple architectures with single layers and the weights are learnt by iteratively changing them until a termination criterion is satisfied. These learning or weight changing procedures are similar to some used in classical clustering approaches. For example the procedure used in the LVQ is similar to the k-means algorithm.

The SOM is often used because it generates an intuitive two dimensional map of a multidimensional dataset but it generates a suboptimal partition if the initial weights are not properly selected.

There are two major concerns for ANN learning:

- *stability* - no point in the training set changes its class after a finite number of learning steps
- plasticity - the ability of the algorithm to adapt to new data

For stability the learning rate should decrease to zero as iterations proceed but this affects the plasticity.

Shang and Wah [84] describe a global optimization approach for the determination of network weights with layered feed-forward networks and attribute the finding of better local minima to the global search stage.

1.2.5 Mixture Models

In the mixture model approach to cluster analysis the underlying assumption is that the data are drawn from a mixture of an initially specified number k of groups in some proportions $\pi_1, \ldots, \pi_k$. That is, the data come from a population whose distribution is the mixture probability density function

$$f(\boldsymbol{y}; \boldsymbol{\Psi}) = \sum_{i=1}^{k} \pi_i c_i(\boldsymbol{y}; \boldsymbol{\theta_i})$$

where the k components correspond to the k groups. Here the vector Ψ of unknown parameters consists of the mixing proportions π_i and the elements of the θ_i known *a priori* to be distinct. On specifying a parametric form for each component p.d.f $c_i(\boldsymbol{y};\boldsymbol{\theta_i})$, Ψ can be estimated by maximum likelihood or some other method. Once the mixture model has been fitted, a probabilistic clustering of the data into k clusters can be obtained in terms of the fitted posterior probabilities of component membership for the data. An outright assignment of the data into the k clusters is achieved by assigning each data point to the component to which it has the highest estimated posterior probability of belonging.

Most of the work in this area has assumed that the individual components of the mixture density are Gaussian and in this case the parameters of the individual Gaussians are to be estimated. More recently, the Expectation Maximization (EM) algorithm has been applied to this problem of parameter estimation [65].

1.2.6 Evolutionary Approaches

One of the most well known heuristic approaches to global optimization is the genetic algorithm. Genetic algorithms (GAs) [49, 44] are an example of a set of approaches known as evolutionary approaches motivated by the ideas of natural selection and evolution. These approaches use evolutionary operators such as *selection, recombination* and *mutation* on a population usually encoded as *chromosomes* to obtain a globally optimal partition of the dataset. In GAs the selection operator propagates solutions from one generation to the next based on their fitness. Selection operators usually use a probabilistic scheme where chromosomes with higher fitness have a higher probability of being reproduced in the next generation.

Points in the search space (chromosomes) are represented as bit strings. The crossover operator as a form of recombination exchanges subsequences of these bit strings between parents and is effective at exploring the search space and mutation acts as a fine tuning for the exploration process. GAs and other evolutionary strategies have been used to solve the clustering problem by viewing it as a minimization of the squared error criterion. Theoretical issues of convergence of these approaches are discussed in Fogel [42].

Whereas most traditional clustering techniques (such as k means, c-means and ANNs) perform a local search to optimize the objective function GAs perform a global search. One of the main problems in the use of GAs in clustering is the representation of the problem using bit strings. There have been many approaches (see Raghavan and Birchand [74], Bhuyan et al. [32] and Jones and Beltramo [55]) to the representation and crossover problems but most restrict the use of GAs on practical datasets to a small number of clusters. Representations which are of low order and short in defining length are required. Babu and Murty [7] implement a hybrid approach where the GA is used to find good initial cluster centres and the k-means algorithm is used to find the final partition.

Another problem with GAs is their sensitivity to the selection of various parameters such as the population size and the mutation and crossover probabilities. This problem has been studied (for example see Grefenstette [45]) but there are still difficulties in achieving good results on specific problems. There have been claims that hybrid genetic algorithms incorporating problem specific heuristics are effective for clustering [55].

In many cases GA approaches perform better than k-means or c-means. However, all of these approaches suffer from sensitivity to the selection of the control parameters. For each specific

problem, there needs to be fine tuning of the parameter values to suit the application.

1.2.7 Fuzzy Clustering

Traditional or hard clustering approaches generate partitions or groups where each data point belongs to one and only one cluster. Fuzzy clustering extends the idea into the *multi-label* domain where data points may be simultaneously in many clusters. Fuzzy clustering extends this notion to associate each data point with with every cluster using a membership function. The output of these algorithms is therefore a clustering rather than a partition. A fuzzy algorithm would usually select an initial fuzzy partition of the n data points into k clusters by initialising the $n \times k$ membership matrix $\boldsymbol{U}$. Compute the value of a fuzzy objective function such as

$$f(\boldsymbol{U}) = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij} \|\boldsymbol{x_i} - \boldsymbol{c_j}\|^2$$

where $c_j = \sum_{i=1}^{n} u_{ij} \boldsymbol{x_i}$ is the j^th fuzzy cluster centre and reassign data points to clusters to reduce this objective function.

In a fuzzy clustering of data, larger values of the membership function for particular clusters indicate higher confidence in the assignment of that data point to that cluster. A hard clustering can be obtained from a fuzzy clustering by deciding on threshold values for membership values. A common fuzzy clustering algorithm is the fuzzy c-means (FCM) algorithm. It is usually better at avoiding local minima, than the k-means algorithm but can still converge to local minima of the squared error criterion. The design of the membership functions is an important problem in fuzzy clustering.

1.2.8 A simulated Annealing Approach

The simulated annealing approach is a sequential stochastic search technique modelled on the annealing of crystals. Simulated annealing algorithms are designed to avoid and recover from solutions which are local minima of the objective function. The technique achieves this by accepting with some probability a new solution of lower quality for the next iteration. The probability of acceptance is governed by a parameter called the temperature which is varies from a starting value at the first iteration to a final value at the final iteration. A simulated annealing clustering algorithm randomly selects an initial partition of the dataset, values for the initial and final temperatures and computes the squared error. Points are reassigned to clusters on the basis of the square error value either with a temperature dependent probability or not. At each iteration the value of the temperature is reduced.

Simulated annealing can be slow to reach an optimal solution. Optimal solutions require the temperature to be decreased very slowly over iterations. Selim and Al-Sultan [82] have studied the effects of the control parameters. Simulated annealing is statistically guaranteed to find the global optimal solution [2].

1.3 Data Classification

Classification is the supervised assignment of data points to predefined and known classes. Here, there is a collection of classes with labels and the problem is to label a new observation

or data point as belonging to one or more of the classes. The objective in many cases (where the data can be represented as real valued) is to learn a function $f : R^n \rightarrow y_i : i = 1 \ldots n$ from the training data. Here the known classes of examples in the training set are represented by the labels y_i. This function can then be used to assign new examples to classes. The problem is to estimate f so that the predicted label $f(x)$ is the true label y for examples $x, y)$ which were generated from the same underlying probability distribution. The problem as represented above determines a classification function or *classifier* f that predicts many classes but only assigns a single class to each data point. There are classification problems that require a fuzzy prediction and these are usually termed multi-label classification problems. Furthermore classifiers are usually binary but the more general multi-class problem can be solved by the learning of several binary classifiers.

Convex programming technique also can be applied for data classification. We consider here only support vector machine technique.

The field of data classification is very large and covers a broad range of areas including biology, information science and bio-informatics. A good review of machine learning, neural and statistical approaches including decision tree algorithms such as C4.5 can be found in Michie et al [67].

1.3.1 Support Vector Machines

From statistical learning theory it is crucial that the class of classifier functions must be restricted to one with an appropriate capacity for the available training data. One such measure of capacity is the *VC dimension*- a measure of the function complexity. For a classifier to generalize from the training data to unseen examples it should be from a class of functions whose capacity can be computed and the algorithm should keep the capacity low as well as fit the training data.

Support vector machine classifiers are based on the class of hyperplanes

$$\langle w, x \rangle + b = 0$$

$w, x \in R^n, b \in \mathbb{R}$ corresponding to binary decision functions

$$f(x) = \text{sign}\,(\langle w, x \rangle + b).$$

It is possible to prove that the optimal hyperplane, defined as the one with the maximal margin of separation between the two classes comes from the function class with the lowest capacity. This hyperplane can be constructed by solving a constrained quadratic optimization problem whose solution has w has an expansion in terms of a subset of the training data that that lie closest to the boundary. These training points, called *support vectors*, carry all relevant information about the classification problem. It is simple to show that the margin is inversely proportional to $\|w\|$ and that the problem is:

$$\text{minimize}(\langle w, w \rangle) \text{ subject to } y_i[\langle w, x_i \rangle + b] \geq 1.$$

The final decision function can be written as

$$f(x) = \langle w, x \rangle + b = \sum y_i \alpha_i \langle x_i, x \rangle + b,$$

where the index i covers only the support vectors. That is, if all data points other than the support vectors were removed, the algorithm would find the same solution. This property of *sparseness* is important in the implementation and analysis of the algorithm. The quadratic programming problem and the final decision function depend only on inner products between data and this permits the generalization of this approach to the nonlinear case via kernel functions.

The main idea underlying kernel methods is the embedding of the data into a higher dimensional vector space. This allows the use of linear algebra techniques and geometry to detect structure in the data [80].

SVMs have achieved impressive results in classification tasks including text categorization [54], handwriting digit recognition [36] and with gene expression data for gene function prediction [31]. They have out performed most other approaches in most problem areas.

2 Cluster analysis via nonsmooth optimization

2.1 Introduction

Clustering is the *unsupervised* classification of patterns.

In cluster analysis we assume that we have been given a set A of a finite number of points of n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1, \ldots, a^m\}, \text{ where } a^i \in \mathbb{R}^n, \ i = 1, \ldots, m.$$

There are different types of clustering such as partitional, packing, covering and hierarchical clustering. In this paper we will consider partitional clustering.

The subject of cluster analysis is the partition of the set A into a given number k of overlapping or disjoint subsets $A^i, \ i = 1, \ldots, k$ with respect to predefined criteria such that

$$A = \bigcup_{i=1}^{k} A^i.$$

The sets $A^i, \ i = 1, \ldots, k$ are called clusters.

There exist different approaches to clustering including agglomerative and divisive hierarchical clustering algorithms as well as algorithms based on mathematical programming techniques. Descriptions of many of these algorithms can be found, for example, in [40, 50, 86]. An excellent up-to-date survey of existing approaches is provided in [51] and a comprehensive list of literature on clustering algorithms is available in this paper.

The clustering problem is said to be *hard clustering* if every data point belongs to one and only one cluster. Unlike hard clustering, the clusters are allowed to overlap in the *fuzzy clustering* approach. In this paper we generally consider the hard unconstrained clustering problem, that is, we additionally assume that

$$A^i \bigcap A^j = \emptyset, \quad \forall i, j = 1, \ldots, k, \ i \neq j.$$

and no constraints are imposed on the clusters $A^i, \ i = 1, \ldots, k$. Thus every point $a \in A$ is contained in exactly one and only one set A^i.

Many authors reduced the clustering problem to the following optimization problem (see, for example, [25, 26, 86]):

$$\text{minimize} \;\; \varphi(\mathcal{C}, x) := \frac{1}{m} \sum_{i=1}^{k} \sum_{a \in A^i} \|x^i - a\|_2^2 \tag{2.1}$$

$$\text{subject to} \;\; \mathcal{C} \in \overline{\mathcal{C}}, \;\; x = (x^1, \ldots, x^k) \in \mathbb{R}^{n \times k}, \tag{2.2}$$

where $\|x\|_2$ is the Euclidean norm, $\mathcal{C} = \{A^1, \ldots, A^k\}$ is a set of clusters, $\overline{\mathcal{C}}$ is a set of all possible k-partitions of the set A and x^i is the center of the cluster A^i, $i = 1, \ldots, k$. (We shall discuss the notion of the *system* of centres of clusters later on.) The following problem also can be considered instead of (2.1)–(2.2):

$$\text{minimize} \;\; \varphi_1(\mathcal{C}, x) := \frac{1}{m} \sum_{i=1}^{k} \sum_{a \in A^i} \|x^i - a\| \tag{2.3}$$

$$\text{subject to} \;\; \mathcal{C} \in \overline{\mathcal{C}}, \;\; x = (x^1, \ldots, x^k) \in \mathbb{R}^{n \times k}. \tag{2.4}$$

The problem (2.3)–(2.4) depends on the choice of a norm: different norms can lead to different centers of clusters. Since clustering is a "flexible" notion, the use of different norms is acceptable.

The following form of (2.1)–(2.2) is more suitable for the application of optimization techniques:

$$\text{minimize} \;\; g(x, w) := \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} w_{ij} \|x^j - a^i\|^2 \tag{2.5}$$

subject to

$$x = (x^1, \ldots, x^k), \quad x^j \in \mathbb{R}^n, \;\; j = 1, \ldots, k, \tag{2.6}$$

and

$$\sum_{j=1}^{k} w_{ij} = 1, \;\; i = 1, \ldots, m, \qquad w_{ij} = 0 \text{ or } 1, \;\; i = 1, \ldots, m, \;\; j = 1, \ldots, k, \tag{2.7}$$

where k is the number of clusters (given), m is the number of available patterns (given), $x^j \in \mathbb{R}^n$ is the j-th cluster's center (to be found), w_{ij} is the association weight of pattern a^i with cluster j (to be found), given by

$$w_{ij} = \begin{cases} 1 & \text{if pattern } i \text{ is allocated to cluster } j \;\; \forall i = 1, \ldots, m, \;\; j = 1, \ldots, k, \\ 0 & \text{otherwise.} \end{cases}$$

Here w is an $m \times k$ matrix. This is a mixed problem (it contains both continuous and integer variables).

The objective function $g(x, w)$ of (2.5) has many local minima. Different methods of mathematical programming can be applied to solve this problem. Some review of these algorithms can be found in [47] with dynamic programming, branch and bound, cutting planes and the k-means algorithms being among them. The dynamic programming approach can be effectively applied to the clustering problem when the number of instances $m \leq 20$, which means that this method is not effective for solving real-world problems (see [53]). Branch and bound algorithms are effective when the database contain only hundreds of records and the number of clusters is not large (less than 5) (see [39, 46, 47, 59]). For these methods the solution

of large clustering problems is out of reach. This leads to the usage of local techniques and different heuristics for solving large clustering problems. One of the popular techniques is the well-known k-means algorithm, which serves for the search of local minima of a problem that is equivalent to (2.1)–(2.2).

Much better results have been obtained with metaheuristics for global optimization, such as simulated annealing, Tabu search and genetic algorithms [75]. The simulated annealing approaches to clustering have been studied, for example, in [30, 82, 87]. Application of tabu search methods for solving clustering is studied in [1]. Genetic algorithms for clustering have been described in [75]. The results of numerical experiments, presented in paper [5] show that even for small problems of cluster analysis when the number of entities $m \leq 100$ and the number of clusters $k \leq 5$ these algorithms take 500-700 (sometimes several thousands) times more CPU time than the k-means algorithms. For relatively large databases one can expect that this difference will increase. This makes metaheuristic algorithms of global optimization ineffective for solving many clustering problems.

An approach to clustering analysis based on bilinear programming techniques has been described by Mangasarian in [63]. If a polyhedral distance, such as the 1-norm distance, is used, the cluster analysis problem can be formulated as that of minimizing the product of two linear functions on a set determined by satisfying a system of linear inequalities. The k-median algorithm consisting of solving a few linear programs leads to a stationary point.

The paper [16] describes a global optimization approach to clustering and demonstrates how the supervised data classification problem can be solved via clustering. A detailed presentation of the main ideas from [16] can be found in [18]. The objective function in the problem from [18] is both nonsmooth and nonconvex and this function has a large number of local minimizers. Problems of this type are quite challenging for general-purpose global optimization techniques. Due to the large number of variables and the complexity of the objective function, general-purpose global optimization techniques as a rule fail to solve such problems.

Objective functions of optimization problems that are equivalent to (2.1)–(2.2) usually have very many shallow local minima, which do not provide a good description of the dataset. However, as mentioned above, global optimization techniques are highly time-consuming. It is very important, therefore, to develop clustering algorithms based on optimization techniques that compute "deep" local minimizers. Such minimizers provide a good enough description of the dataset under consideration. Indeed, since clustering is a flexible notion, a deep local minimizer may satisfactory describe the clustering structure of this dataset. A different approach, which can be successfully used for supervised classification, is to change the setting of the optimization problem under consideration and consider a series of simpler problems (step-by-step approach). Some versions of this step-by-step approach can also be used for clustering.

Datasets under consideration usually contains two types of features (coordinate of a vector): continuous and categorical. Continuous features usually reflect results of some measurements. A feature is called categorical, if it is either nominal or ordinal. A variable is a "nominal" one if its values only numerical codes of possible different states of the corresponding feature. If the states of a nominal variable can be arranged in a meaningful order, the term "ordinal variable" is used.

Our experience demonstrates that optimization algorithms work much better when a dataset contains only vectors with continuous features. However optimization algorithms can also be

used for some datasets with categorical features (see, for example, Subsection 3.10).

2.2 The nonsmooth optimization approach to clustering

Here we describe an approach to clustering from [18]. This approach leads to a nonsmooth and non-convex optimization problem, which is equivalent to (2.3)– (2.4), but simpler from the computational point of view.

We consider an n-dimensional space $\mathbb{R}^n$ equipped with a norm $\|\cdot\|$. As a rule we assume that $\|\cdot\| = \|\cdot\|_p,\ p \geq 1$, where

$$\|x\|_p = \left(\sum_{l=1}^{n} |x_l|^p\right)^{1/p}, \ \ 1 \leq p < +\infty.$$

The l-th coordinate of a vector $x \in \mathbb{R}^n$ is denoted by x_l.

Consider a set A of m n-dimensional vectors $a = (a_1, \ldots, a_n)$. The aim of clustering is to represent this set as the union of k clusters. We accept the hypothesis that each cluster can be described by a point that can be considered as its center. So we need to locate a cluster's center in order to adequately describe the cluster itself. Thus, we would like to find k points that serve as centers of k clusters $A^1, \ldots, A^k$. First we need to give the formal definition of the centres of a finite system of finite mutually disjoint sets $A^1, \ldots, A^k$, having in mind that these sets are unknown and we know only their union.

Consider an arbitrary set X, consisting of k points $x^1, \ldots, x^k$. The distance $d(a, X)$ from a point $a \in A$ to this set is defined by

$$d(a, X) = \min_{s=1,\ldots,k} \|x^s - a\|.$$

The deviation $d(A, X)$ from the set A to the set X can be calculated as

$$d(A, X) = \sum_{a \in A} d(a, X) = \sum_{a \in A} \min_{s=1,\ldots,k} \|x^s - a\|.$$

We say that the set $\bar{X} = (\bar{x}^1, \ldots, \bar{x}^k)$ is the set of centres of k clusters of the set A if $d(A, \bar{X}) \leq d(A, X)$ for each $X = (x^1, \ldots, x^k)$. Thus the search for centres of clusters (hence, the search for the clusters) can be reduced to the following unconstrained minimization problem:

$$\text{minimize} \ \ C_k(x^1, \ldots, x^k) \quad \text{subject to } (x^1, \ldots, x^k) \in \mathbb{R}^{n \times k}, \tag{2.8}$$

where

$$C_k(x^1, \ldots, x^k) = \frac{1}{m} \sum_{a \in A} \min_{s=1,\ldots,k} \|x^s - a\|. \tag{2.9}$$

The function C_k defined by (2.9) will be called the *cluster function*. Figure 1 illustrates a plot of the cluster function C_2 in $\mathbb{R}^1$ for a dataset with 20 points.

If $k > 1$, the cluster function is nonconvex and nonsmooth. It can be shown that this function has very many shallow local minimizers, which are close each to other.

Note that the number of variables in the optimization problem (2.8) is $k \times n$. If the number k of clusters and the number n of attributes are large, we have a large-scale global optimization

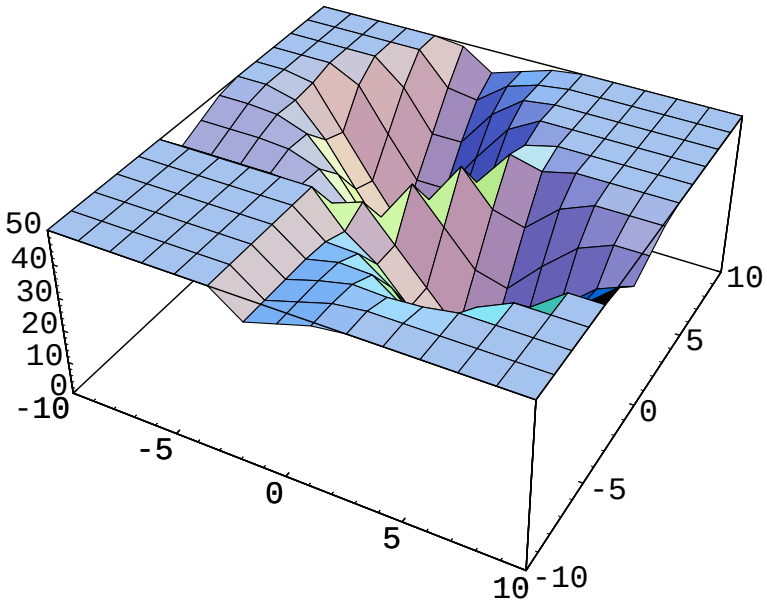

Figure 1: Cluster function in $\mathbb{R}^2$

problem. Moreover, the form of the objective function in this problem is complex enough not to be amenable to the direct application of general-purpose global optimization methods. Therefore, in order to ensure the practicality of the nonsmooth optimization approach to clustering, proper identification and use of local optimization methods is very important.

Clearly, such an approach does not guarantee a globally optimal solution to problem (2.8). However, because clustering is a flexible notion, we do not need to obtain the exact solution of (2.8), it is enough to have a good approximation of this solution. This approximation can be accomplished by a "deep" enough local minimum of the cluster function. We can suppose that a deep local minimizer provides a good enough clustering description of the dataset under consideration. Thus we will be concerned with the search for "deep" local minima. The local method for non-smooth optimization that we use (see Section 4) can avoid saddle points and even some shallow local minimizers. Combination of this method with some global techniques can help to attain a deep local minimizer.

The following version of the problem (2.8) can be considered:

$$\text{minimize } \frac{1}{m}\sum_{a\in A}\min_{s=1,\ldots,k}\|x^s-a\|^2 \quad \text{subject to } (x^1,\ldots,x^k)\in \mathbb{R}^{n\times k}, \tag{2.10}$$

where $\|\cdot\|$ is 2-norm.

It is shown in [25] that problems (2.1)–(2.2), (2.5–2.7) and (2.10) are equivalent. The number of variables in problem (2.5)–(2.7) is $(m+n)\times k$ whereas in problem (2.10) this number is only $n\times k$ and the number of variables does not depend on the number of instances. It should be noted that in many real-world databases the number of instances m is substantially greater than the number of features n. On the other hand the coefficients w_{ij} are integer in hard clustering problems, that is the problem (2.5)–(2.7) contains both integer and continuous variables. We have only continuous variables in the nonsmooth optimization formulation

of the clustering problem. All these circumstances can be considered as advantages of the nonsmooth optimization formulation (2.8) and its version (2.10).

2.3 Cluster function as a tool for measuring the fitness of a collection of points

Very often we have different candidate vectors $X = (x^1, \ldots, x^k)$ which contend to be considered as centres of clusters. It is an important task to compare these candidates. We now describe the simplest situation where we need such a comparison. Assume that we use a certain local method for the search for centers of clusters. We can run this method many times from different initial points. As a rule we obtain different results. How is the best among them chosen?

We can use the cluster function for the comparison. Assume that a norm $\|\cdot\|$ is fixed and we consider the cluster function f generated by this norm. It follows directly from the definition of the cluster function, that the candidate $X_* = (x^1_*, \ldots, x^k_*)$ is better suited for the role of centres of clusters than candidate $X = (x^1, \ldots, x^k)$ in the sense of the norm $\|\cdot\|$ if $C_k(x^1_*, \ldots, x^k_*) < C_k(x^1, \ldots, x^k)$.

2.4 k-means algorithm

The k-means algorithm is one of the effective algorithms for solving clustering problems on large datasets. Different versions of this algorithm have been studied by many authors (see [86]). This algorithm is based on the the minimization of the variation within clusters and the maximization of the variation between clusters. The variation depends on the chosen norm. Usually the norm $\|\cdot\|_2$ is used for this purpose, however versions of k-means with different norms also can be used. For simplicity we describe the simplest version of this method (see, for example, [69]) which was developed by MacQueen in 1967 (see [61]). In this paper we use this version and also its modification, where $\|\cdot\|_1$ is used instead of $\|\cdot\|_2$.

The method starts with a user-specified value of k (number of clusters) points. To sort m observations into k clusters with a given norm $\|\cdot\|$, we use the following procedure.

1. Take any k observations as the centers of the first k clusters.
2. Assign the remaining $m - k$ observations to one of the k clusters on the basis of the shortest distance (in the sense of the norm we choose) between the observation and the center of the cluster.
3. After each observation has been assigned to one of the k clusters, the centers are recomputed (updated) as the centroids of found clusters.

Stopping criteria: there is (almost) no observation, which moves from one cluster to another.

k-means is a very fast algorithm and it is suitable for solving clustering problems in large databases. This algorithm gives good results when there are only a few clusters but deteriorates when there are many [47]. If $\|\cdot\|_2$ is used then k-means achieves a local minimum of problem (2.1)(see [81]), however experiments show that the best clustering found with

k-means may be more than 50 % worse than the best known one [47]. This is because, the k-means, like the majority of local methods, is very sensitive to the choice of the initial point. These experiments shows that k-means usually leads to a shallow local minimum of (2.1), which does not describe the cluster structure well.

2.5 An optimization clustering algorithm

A meaningful choice of the number of clusters is very important for clustering analysis. It is difficult to define *a priori* how many clusters represent the set A under consideration. In order to increase the knowledge generating capacity of the resulting clusters, the decision maker has to start from a small enough number of clusters k and to gradually increase the number of clusters for the analysis until certain termination criteria motivated by the underlying decision making situation as satisfied. From an optimization perspective this means that if the solution of the corresponding optimization problem (2.8) is not satisfactory, the decision maker needs to consider problem (2.8) with $k+1$ clusters and so on. This implies that one needs to solve repeatedly arising global optimization problems (2.8) with different values of k - a task even more challenging than solving a single global optimization problem. In order to avoid this difficulty, we suggest a step-by-step calculation of clusters.

The main idea of the proposed algorithm (see [22]) is to use the results obtained at a certain step for finding a good initial state for the next step. Note the cluster function for $k = 1$ is convex, so we can use convex programming techniques if $k = 1$.

The proposed approach has two distinct important and useful features:

- it allows the decision maker to successfully tackle the complexity of large datasets as it aims to reduce the number of data instances (records) of the dataset under consideration without loss of valuable information
- it provides the capability of calculating clusters step-by-step, gradually increasing the number of data clusters until termination conditions are met, that is it allows one to calculate as many cluster as a dataset contains with respect to some tolerance.

Algorithm 2.1 An algorithm for solving a cluster analysis problem.

Step 1. (Initialization). Select a tolerance $\varepsilon > 0$ and an positive integer k_0 as the starting number of clusters. Select a starting point $x^0 = (x_1^0, \ldots, x_n^0, \ldots, x_1^{k_0}, \ldots, x_n^{k_0}) \in \mathbb{R}^{n \times k_0}$ and solve the minimization problem (2.8) with $k = k_0$. Let $x^{1*} \in \mathbb{R}^{n \times k_0}$ be a solution to this problem and C_{1*} be the corresponding objective function value. Set $k = k_0$.

Step 2. (Computation of the next cluster center). Select a point $x^0 \in \mathbb{R}^n$ and solve the following minimization problem of the dimension n:

$$\text{minimize} \quad \bar{C}_k(x) \quad \text{subject to} \quad x \in \mathbb{R}^n \tag{2.11}$$

where

$$\bar{C}_k(x) = \sum_{a \in A} \min \left\{ \|x^{1*} - a\|, \ldots, \|x^{k*} - a\|, \|x - a\| \right\}.$$

Step 3. (Refinement of all cluster centers). Let $\bar{x}^{k+1,*}$ be a solution to problem (2.11). Take $x^{k+1,0} = (x^{1*}, \ldots, x^{k*}, \bar{x}^{k+1,*})$ as a new starting point and solve (2.8) with the objective function C_{k+1}.

Step 4. (Stopping criterion). Let $x^{k+1,*}$ be a solution to the problem (3.4) and $C_{k+1,*}$ be the corresponding value of the objective function. If

$$\frac{C_{k*} - C_{k+1,*}}{C_{1*}} < \varepsilon$$

then stop, otherwise set $k = k + 1$ and go to Step 2.

In Step 1 the centers of the first k_0 clusters are calculated. In particular, one can take $k_0 = 1$, then the center of the entire set A will be calculated. In Step 2 we calculate a center of next $(k+1)$-st cluster, assuming previous k cluster centers to be known. Note that the number of variables in problem (2.11) is n which is substantially less the number of variables when all cluster centers are calculated simultaneously. In Step 3 the refinement of all $k+1$ cluster centers is carried out. It is quite possible that the starting point $x^{k+1,0}$ calculated in the previous Step 2 is not far from the solution to problem (2.8), so it takes only a moderate number of iterations to calculate this solution. Such an approach allows significant reduction in the computational time for solving problem (2.8).

It is clear that $C_{k*} \geq 0$ for all $k \geq 1$ and the sequence $\{C_{k*}\}$ is decreasing:

$$C_{k+1,*} \leq C_{k,*} \quad \text{for all} \quad k \geq 1.$$

Hence after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

One of the important questions when one tries to apply Algorithm 2.1 is the choice of the tolerance $\varepsilon > 0$. Large values of ε can result the appearance of large clusters whereas small values can produce small and artificial clusters. In order to explain this, let us consider an artificial dataset on $\mathbb{R}^2$ as shown in Figure 2. There are three isolated clusters in this dataset given by the following formulae, respectively:

$$A^1 = \left\{ a^k \in \mathbb{R}^2 : a^k = (a_1^k, a_2^k), a_1^k = \frac{1}{2}|\sin(k)|,\ a_2^k = 2 + |\cos(k)|,\ k = 1, \ldots, 50 \right\},$$

$$A^2 = \left\{ a^k \in \mathbb{R}^2 : a^k = (a_1^k, a_2^k), a_1^k = \frac{1}{2}(1 + |\sin(k)|),\ a_2^k = |\cos(k)|,\ k = 1, \ldots, 50 \right\},$$

$$A^3 = \left\{ a^k \in \mathbb{R}^2 : a^k = (a_1^k, a_2^k), a_1^k = 2 + |\cos(k)|,\ a_2^k = 1.5 + \sin(k),\ k = 1, \ldots, 50 \right\}.$$

If $\varepsilon = 10^{-1}$ then Algorithm 2.1 exactly calculates these three clusters, if $\varepsilon = 10^{-2}$ then this algorithm divides the third cluster into three clusters. When ε is smaller we have further division of these clusters. So if ε is small enough we obtain some artificial clusters. The results of numerical experiments show that the best values for ε are $\varepsilon \in [10^{-1}, 10^{-2}]$.

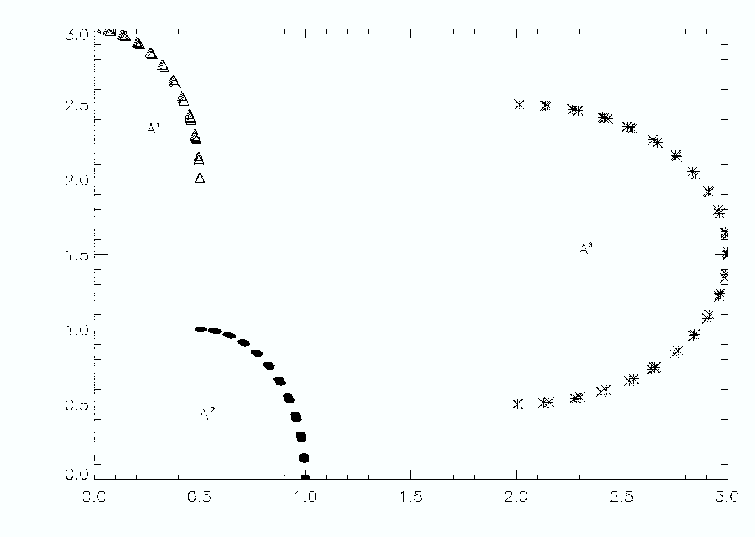

Figure 2: Three clusters in $\mathbb{R}^2$

2.6 Results and discussion

To verify the effectiveness of the clustering algorithm a number of numerical experiments with middle-sized and large datasets have been carried out on a Pentium-4, 1.7 GHz, PC.

First we consider three standard test problems to compare our algorithm with k-means and metaheuristics: the tabu search (TS) method, a genetic algorithm (GA) and a simulated annealing (SA) method. We use the results obtained using these algorithms and presented in [5] for comparison. These methods have been applied to problem (2.5)– (2.7) which is equivalent to (2.10).

We have used the well-known 'German towns" and two 'Bavaria postal zones' test datasets (see Appendix).

Results of the numerical experiments are presented in Tables 1-3. In these tables we give the values of local (and possible global) minima obtained by the different algorithms for different number of clusters. These values are given as $mf(x^*)$ where m is the number of instances and x^* is a local minimizer.

Table 1: Results for German towns database

Number of clusters	k-means	TS	GA	SA	Algorithm 2.1
2	121425.75	121425.75	121425.75	121425.75	121425.75
3	78127.50	77008.62	77008.62	77233.67	77233.67
4	51719.90	49600.59	49600.59	49600.59	49600.59
5	40535.70	39452.81	39511.05	39511.05	39511.05

The results presented in Table 1 show that Algorithm 2.1 achieves better results than k-means

for all number of clusters. The results from this algorithm and SA are similar. Tabu search is better than Algorithm 2.1 for three and five clusters, however the results are close. The genetic algorithm is slightly better than Algorithm 2.1 for three clusters. We can see that Algorithm 2.1 finds solutions which are similar or very close to the solutions obtained by the global optimization techniques. This means that this algorithm can calculate "deep" local minima of the objective function in a clustering problem.

Table 2: Results for the first Bavarian postal zones dataset.

Number of clusters	k-means	TS	GA	SA	Algorithm 2.1
2	6.49245E11	6.02546E11	6.02547E11	4.63008E11	6.02547E11
3	3.63652E11	3.63652E11	3.63653E11	3.63653E11	2.94507E11
4	2.78805E11	1.23421E11	1.04475E11	1.04873E11	1.04475E11
5	2.60158E11	7.96901E10	5.97614E10	8.38579E10	5.97614E10

From Table 2 we can see that Algorithm 2.1 again gives better results than the k-means algorithm. This algorithm achieves the best results for all values of k, except $k = 2$.

Table 3: Results for the second Bavarian postal zones dataset.

Number of clusters	k-means	TS	GA	SA	Algorithm 2.1
2	4.86313E10	1.99080E10	4.86313E10	4.86313E10	4.86313E10
3	3.59792E10	1.73987E10	1.73987E10	3.09296E10	1.73987E10
4	3.05136E10	7.55908E9	7.55908E9	8.24909E9	7.55908E9
5	2.95115E10	6.25550E9	6.25560E9	6.41519E9	5.40379E9

The results presented in Table 3 show that for this dataset, Algorithm 2.1 achieves better results than the k-means algorithm. Moreover this algorithm gives the best results for $k = 3, 4, 5$.

Based on the results presented in Tables 1-3 we can conclude that at least for these three datasets, Algorithm 2.1 works better than the k-means algorithm and achieves close, similar and sometimes better results than tabu search, genetic and simulated annealing algorithms, using significantly less CPU time. These results confirm that the proposed algorithm, as a rule, finds "deep" local minima or sometimes a global minimum of the objective function in a clustering problem.

Finally, we applied our algorithm to calculate clusters in some databases with known classes. We used the diabetes, liver disorder, heart disease, breast cancer, vehicles, synthetic, pen-based recognition of handwritten digits (PBRHD) and image segmentation datasets in numerical experiments. Descriptions of these datasets can be found in Appendix.

First, we normalized all features. This was done by a nonsingular matrix so that mean values of all features were 1.

In numerical experiments we take $\varepsilon = 10^{-2}$ and the initial number of clusters $k_0 = 2$. First

we applied Algorithm 2.1 to calculate clusters. Then the k-means algorithm was applied with the same number of clusters as calculated by Algorithm 2.1.

Results of the numerical experiments are given in Table 4. In this table we present the characteristics of a dataset where m is the number of instances, n number of attributes (features), k the number of classes. We also present the accuracy and the value of objective function achieved by Algorithm 2.1 and k-means. In situations where instances are already labelled, we can compare the clusters with the "true" class labels. We use the notion of cluster *purity* defined in [38] as:

$$P(C) = \frac{1}{n_C} \max_{i=1,\ldots,l} \quad n_C^i$$

to evaluate accuracy. In this expression $n_C = |C|$ is the cardinality of the cluster C, n_C^i is the number of instances in the cluster C that belong to class i and l is the number of classes.

Table 4: Results of clustering in databases with known classes

Database	$m \times n \times k$	$n_{cluster}$	Algorithm 2.1		k-means	
			Accuracy	Obj. func.	Accuracy	Obj. func.
Diabetes	$768 \times 8 \times 2$	13	68.2	1.0405	64.7	1.0824
Heart	$297 \times 13 \times 2$	8	73.1	3.0691	75.2	3.1902
Liver	$345 \times 6 \times 2$	10	64.3	0.5405	63.6	0.5488
Br. cancer	$683 \times 9 \times 2$	6	97.5	1.9650	96.5	2.1534
Synthetic	$600 \times 60 \times 6$	9	88.1	1.4219	86.7	1.5253
Vehicles	$846 \times 18 \times 4$	14	49.4	0.3896	47.1	0.3943
PBRHD	$10992 \times 16 \times 10$	12	79.9	1.9684	83.0	1.9810
Image seg.	$4435 \times 36 \times 6$	10	83.0	0.2817	81.5	0.2875

The results presented in Table 4 show that Algorithm 2.1 gives better results for all datasets, except the vehicles dataset where the results are almost similar. We can see that, as a rule, the smaller the value of the objective function the better the description of the dataset. However, this is not true always but in these cases the difference is very small. Note that the values of the objective function appearing in the table are given after scaling and dividing by the number of instances. The effect of this is that the actual differences are much larger than appearing in the table.

Remark 2.1 Often it is implicitly assumed that classes coincide with some clusters (or the union of some clusters) in all datasets with known classes. Of course this is only an assumption that should be checked. For some datasets the accuracy of the results of clustering in Table 4 are not very high, which shows that there is now straightforward relation between classes and clusters in these datasets. We shall explain this situation later on (see Subsection 3.10).

2.7 The k-means algorithm and the minimization of the cluster function

Most local methods are very sensitive to the choice of the initial point. In some cases it is reasonable to consider the result, obtained by one local method, as the initial point for another one. Since the k-means method is very fast, very often it used for initialization

Table 5: Medium-size datasets: k-means and optimization

Dataset	Clusters	Features	k-means	Optimization
Vehicle	4	1-18	1996	1814
	15	1-18	1458.8	1293.8
	15	3,4,6-8,11,12,15,16	997	933.66
	15	3,6-8,12,15,16	869	822.6
Liver disorder	6	1-6	494	456
	6	1-5	352	294
	8	1-6	1735	497
	4	1-6	1735.3	614.7
Diabetes	6	1-8	1820	1737
	8	1-8	1709.56	1585
	4	1-8	2047	1908
Australian credit	6	8,9,14	708.7	493
	4	8,9,14	958.5	669
	4	1-14	5754	4743
	4	2,3,5,7,8,13,14	3257	2610
	6	2,3,5,7,8,13,14	2496	2231

of more computationally expensive methods. We consider a combination of k-means with non-smooth local optimization (Discrete gradient method, see Section 4).

For testing the efficiency of the combination of k-means and Discrete Gradient method, we use four well-known medium-size test datasets: Australian credit dataset, Diabetes dataset, Liver disorder dataset and Vehicle dataset. The description of these datasets can be found in Appendix. We studied these datasets, using different subsets of features and different numbers of clusters and without any division of them into classes.

We calculated the value of the cluster function for initial points x obtained by the k-means algorithm and then for the points $\bar{x}$ obtained by Discrete Gradient method starting from x. The results are shown in Table 5. The second column in this table shows the number of clusters, the features, which are taken into account, are included in the third column, the 4th and 5th columns contain values of cluster function corresponding to points x found by k-means (4th column) and points $\bar{x}$ obtained by Discrete Gradient method starting from x (5th column). We used either all features or some subsets of features that were found by the feature selection procedure, that is described in Subsection 3.3.

We came to the conclusion, that the combination of the k-means method and the Discrete Gradient method often works efficiently and produces better results than the k-means method only. Sometimes this improvement is not so significant, but for the liver disorder dataset (8 clusters and 4 clusters) and the Australian Credit dataset (6 clusters with features 8,9,14) this improvement is good enough.

We also considered the continuation of this process. We ran the k-means method from the initial point, obtained after the nonsmooth optimization method. Empty clusters appeared after the second iteration, and the results were not so good. So, for the medium-size datasets under consideration we found, that it is reasonable to make only one iteration for the combination method.

2.8 Clusters and their structure

A cluster with known centre x^j can be described as the set of points $a \in A$ such that

$$\|x^j - a\| \leq \min_{i \neq j} \|x^i - a\|.$$

Consider points $a \in A$ such that

$$\|x^j - a\| \leq c \min_{i \neq j} \|x^i - a\|$$

where $c > 0$. If c is small enough then we can assign a to the cluster j with the greater confidence. Let

$$\phi_j(a) = \min\{c > 0 : \|x^j - a\| \leq c \min_{i \neq j} \|x^i - a\|\},$$

$$\mu_j(a) = \frac{1}{1 + \phi_j(a)}.$$

Then

$$0 < \mu_j(a) \leq 1, \qquad \mu_j(a) = 1 \iff a = x_j,$$
$$\mu_j(a) \geq 1/2 \iff \|x^j - a\| \leq \min_{i \neq j} \|x^i - a\|.$$

We can consider μ_j as a membership function for the fuzzy cluster j. Thus each point belongs to each cluster in a fuzzy sense. Clearly if $\mu_j(a)$ is closer to one then the confidence that a belongs to the cluster j is greater.

For a better understanding of the structure of the cluster j we need to consider the sets

$$\{a \in A : \|x^j - a\| \leq c_l \min_{i \neq j} \|x^i - a\|\}$$

with different $c_l \leq 1$ (for example $c_l = 1, 0.9, 0.8, \ldots, 0.1$.) Often these sets with small enough c_l are either empty or "almost empty". This can be interpreted in the following way: the cluster j is located in a certain ring; there are (almost) no points of this cluster, which are close to its centre. We can describe this situation by saying that the cluster is concentrated in its periphery and sparse and its centre. Points, which are in the outer periphery, can be considered as questionable. A change of the norm is more likely to lead to the change of the membership for these points.

We can use the structure of clusters in order to estimate the quality of a given candidate $X = (x^1, \ldots, x^n)$ as the centres of clusters. The candidate X_* is more appropriate than X, if clusters corresponding to X_* are more concentrated in the vicinity of their centres. This approach is a complementary to the estimation of centres of clusters by means of cluster function.

The results of numerical experiments demonstrate that the minimization of the cluster function with initial points obtained by k-means algorithm often improves the structure of clusters. We considered the following datasets: emergency, pen-based recognition of handwritten digits, image segmentation and letters dataset (see Appendix for their description). We found the clusters within the datasets without any division into classes. We present here one of the brightest examples (for Emergency dataset with 7 clusters). We used the norm $\|\cdot\|_1$.

Table 6: Emergency dataset. k-means algorithm.

c	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
cluster I	7727	7379	6981	6429	5861	4553	2679	702	18	0
cluster II	471	435	389	330	266	191	103	39	5	0
cluster III	1113	956	781	592	409	235	95	18	1	0
cluster IV	1611	1401	1215	980	695	443	226	61	9	0
cluster V	803	691	584	477	328	200	100	43	4	0
cluster VI	92	88	79	73	68	56	44	26	8	1
cluster VII	3369	2928	2464	1869	1330	799	265	59	1	0

Table 7: Emergency. Discrete gradient algorithm.

c	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
cluster I	4034	3612	2892	2188	1975	1741	1499	1202	777	174
cluster II	921	847	759	614	429	288	183	92	27	12
cluster III	2901	2478	2060	1651	1238	922	701	438	169	14
cluster IV	1584	1372	1194	987	725	496	316	179	76	5
cluster V	1966	1737	1451	1114	785	528	302	146	48	4
cluster VI	123	112	101	94	88	76	56	39	11	2
cluster VII	3657	3268	2477	2067	1849	1559	1290	932	527	111

In Table 6 and Table 7 we present the structure of clusters, obtained by the k-means method and the nonsmooth optimization algorithm, respectively. An initial point for optimization is the set of centers of clusters that were found by the k-means method.

Notice that the Discrete gradient algorithm reached another location for the centres. Many points within the clusters moved from the periphery of the clusters to their centres. The value of the objective function, multiplied by the number of points in dataset was decreased from 22446.3 to 19408.4.

2.9 Generalized cluster functions and complexity reduction for large-scale datasets

Due to the highly combinatorial nature of clustering problems, two characteristics of a given dataset can severely affect the performance of a clustering tool: the number of data records (instances) and the number of data attributes (features). In many cases the development of effective tools requires the reduction of both the number of features and the number of instances without loss of knowledge generating ability. We firstly consider the reduction of the number of instances. The reduction of the number of features will be discussed later on (See Subsection 3.3 and Subsection 3.4).

Large-scale datasets usually contain a huge number of points located in a bounded set. Thus many points from this dataset are very close to each other. Let $A \subset \mathbb{R}^n$ be a finite set. Assume that a certain small neighborhood of a point $b \in \mathbb{R}^n$ contains m_b points from A. We can approximate each of these points by b and replace the corresponding part of the cluster function by the simple term $m_b \min_i \|x_i - b\|$.

More precisely, for a given A and for a given tolerance ε consider a set $B \subset \mathbb{R}^n$, such that

for each $a \in A$ there exists $b \in B$ with the property $\|a - b\| < \varepsilon$. We say that a collection $(A_b)_{b\in B}$ of subsets of A is an ε-disjoint cover of A if

$$\|a - b\| < \varepsilon,\ (a \in A_b), \quad A_b \cap A_{b'} = \emptyset\ (b \neq b'), \quad A = \bigcup_{b\in B} A_b.$$

Let m_b be the cardinality of A_b. Replacing each $a \in A_b$ with b in the presentation of the cluster function C_k we obtain the following function

$$\tilde{C}_k(x^1, \ldots, x^k) = \frac{1}{m} \sum_{b\in B} m_b \min(\|x^1 - b\|, \ldots, \|x^k - b\|),$$

which will be called the *generalized cluster function.*

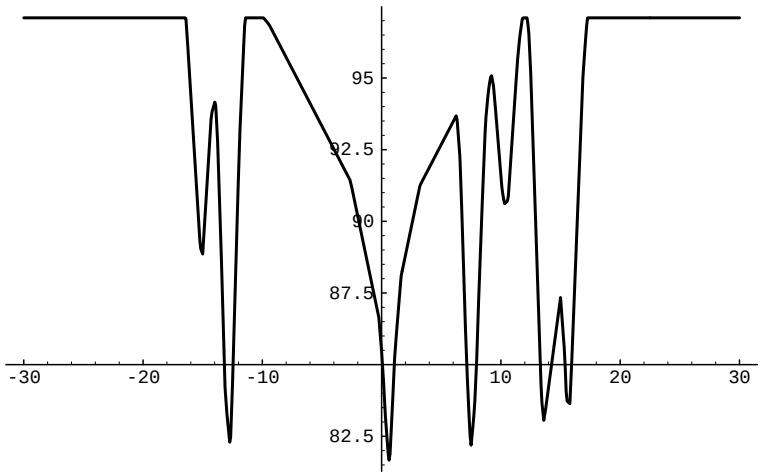

Figure 3: Generalized cluster function in $\mathbb{R}^1$

Figure 3 illustrates a plot of a function of one variable $f(x) = \tilde{C}_2(x, \bar{x}_2)$ with fixed $\bar{x}_2$, where $\tilde{C}_2$ is a generalized cluster function in $\mathbb{R}^1$ for a dataset with 23 points.

We shall use generalized cluster functions for the approximation of cluster functions. Since the notion of cluster is flexible, we can consider an appropriate approximation of the cluster function, which can even be not very exact. The following assertion holds (see [77]).

Proposition 2.1 *Let $(A_b)_{b\in B}$ be an ε-disjoint cover of A and $\tilde{C}_k$ be the generalized cluster function corresponding to this cover. Then $|C_k(x^1, \ldots, x^k) - \tilde{C}_k(x^1, \ldots, x^k)| < \varepsilon$ for all $(x^1, \ldots, x^k) \in (\mathbb{R}^n)^k$.*

Proposition 2.1 allows us to substitute the given dataset A for a smaller dataset B. The minimization of the generalized cluster function $\tilde{C}_k$ corresponding to this set will give us some points $(x^1, \ldots, x^k)$. We can consider these points as centres of k clusters of the set A. Then a cluster A_j corresponding to centre x^j can be described as the union of sets A_b over all b such that $\|b - x^j\| \leq \min_{i\neq j} \|b - x^i\|$.

We now suggest a simple scheme for the construction of an ε-disjoint cover of A with the given tolerance ε. Let $A = \{a_i\}_{i=1,\ldots,m}$ be a given dataset. (Such a representation of the set

A means that we consider this set with a certain order relation: for each point $a \in A$ we indicate its number. The procedure described below depends on the order relation given on the set A. However, as was found from calculations, this dependance is not significant.)

Let $D = (d_{ij}))_{i,j=1,\ldots,m}$ be a symmetric matrix with $d_{ij} = \|a^i - a^j\|$. We shall consider the following procedure: select the first vector a^1, remove from the dataset all the vectors for which $d_{1j} \leq \varepsilon$, and assign to this vector the number m_1 of removed vectors. Denote $a^1 = b^1$. Then select the next remaining vector b^2 and repeat the above procedure for this vector, *etc.* As the result of this procedure we get a subset $B = \{b^j\}_{j=1}^l$ of the given set A and the set $(m_j)_{j=1}^l$, where m_j is the number of removed vectors at the step j. The cardinality l of B can be significantly less then the cardinality m of A. For the search for clusters of the set A we shall apply the generalized cluster functions

$$\tilde{C}_k(x^1, \ldots, x^k) = \frac{1}{m} \sum_{j=1}^{l} m_j \min(\|x^1 - b^j\|, \ldots, \|x^k - b^j\|).$$

In this scheme a selected vector b^j serves as the representative of the set A^j of all points removed at the step j.

Remark 2.2 Different versions of this approach are also possible. For example we can use a point b^j only for the determination of the set A_j. The centroid of this set can be chosen as its representative.

For implementation of the scheme under consideration we need to suggest an appropriate choice of the tolerance ε. We consider two different approaches to this choice.

2.10 Complexity reduction for large-scale datasets: first approach

For each $i,\ i = 1, \ldots, m$, calculate

$$r_i = \min_{j \neq i} d_{ij},$$

where $d_{ij} = \|a_i - a_j\|$. Let

$$r_0 = \frac{1}{m} \sum_{i=1}^{m} r_i.$$

Select $\varepsilon = cr_0$ with $c > 0$.

The results of numerical experiments reported below suggest that such a procedure allows one to significantly reduce the number of instances in the dataset.

In this scheme a key element is the choice of the parameter c. We can give some recommendations on the choice of this parameter using the results of numerical experiments. The dependence of the number of clusters on this parameter can be studied and we use three databases: diabetes, heart disease and liver disorder databases in numerical experiments. The description of these databases can be found in the Appendix. We applied the Algorithm 2.1 to the entire dataset without dividing it into classes. We took the tolerance ε from Step 3 in Algorithm 2.1 equal to 10^{-2} and repeated the computations for different c. Results of numerical experiments are given in Tables 8-10. In these tables we present the number of clusters $n_{cluster}$ calculated by the algorithm, the number of instances remained after application of the scheme for particular value of the parameter c and CPU time.

Table 8: Results of numerical experiments for the diabetes database

c	$n_{cluster}$	Number of instances used/total	CPU time
0.0	13	768/768	274.73
1.0	13	389/768	115.44
1.5	14	283/768	113.64
2.0	13	215/768	66.59
4.0	11	94/768	18.45
6.0	5	52/768	2.31
8.0	5	38/768	2.02

The results presented in Table 8 show that for the diabetes dataset we can take $c \in [0, 4]$. Further decrease of c leads to sharp changes in the cluster structure of the dataset. We can see that there are differences in the number of clusters when $c \in [0, 4]$. But for $c \in [0, 2]$ these differences arise because of small clusters. We can also see that for $c \in [1.5, 4]$ the number of instances and CPU time reduce significantly.

Table 9: Results of numerical experiments for the heart disease database

c	$n_{cluster}$	Number of instances used/total	CPU time
0.0	8	297/297	75.59
1.0	7	152/297	26.73
1.5	6	122/297	14.43
2.0	5	107/297	8.25
4.0	5	65/297	5.05
6.0	5	41/297	3.34
8.0	5	28/297	3.22

The results presented in Table 9 show that appropriate values for the heart disease dataset are $c \in [0, 1.5]$, because further decrease in c leads to changes in the cluster structure of the dataset. We can again see that these values of c allow significant reduction in the number of instances and CPU time.

From the results presented in Table 10 we can conclude that appropriate values of c for the liver disorder dataset are $c \in [0, 2]$. Differences in the number of clusters when $c \in [0, 2]$ arise because of small clusters which contain less than 5 % of all instances.

Thus using results of numerical experiments on these three datasets we can conclude that appropriate values for the parameter c are $c = 0 \in [0, 2]$ and preferable values are $c \in [1.5, 2]$ because in the latter case we can remove the maximum number of instances from a dataset without serious changes in the cluster structure of the dataset.

Table 10: Results of numerical experiments for the liver disorder database

c	$n_{cluster}$	Number of instances used/total	CPU time
0.0	10	345/345	57.66
1.0	12	129/345	37.48
1.5	11	97/345	24.95
2.0	13	79/345	37.16
4.0	8	43/345	4.33
6.0	7	34/345	3.13
8.0	7	29/345	2.67

2.11 Another approach to complexity reduction for large-scale datasets

Let ε be a given tolerance. Proposition 2.1 shows that the generalized cluster function $\tilde{C}_k$ approximates the cluster function C_k with the absolute error ε. Since we are mainly interested in the relative error, it is appropriate to choose $\varepsilon = \varepsilon_h$ where

$$\varepsilon_h = hC_k(x^1, \ldots, x^k)$$

and $x = (x^1, \ldots, x^k)$ is a point close enough to a global minimum. We can choose x as the result of the k-means algorithm starting from a certain initial point. The procedure with such a choice of ε_h will be called the ε_h-cleaning procedure.

Numerical experiments have been carried out with some datasets of medium size and large-scale datasets. We give here only results for the liver disorder, diabetes and heart disease datasets (Table 11) and emergency dataset (Table 12). Descriptions of all these datasets can be found in the Appendix. Here we consider the centroid of the class of the removed points as its representative. (See Remark 2.2.) We use the combination of k-means with the Discrete Gradient method for the minimization of the generalized cluster function. We also use both norms $\|\cdot\|_1$ and $\|\cdot\|_2$. Different norms give different results but they are quite comparable.

The following observation is of a certain interest. Assume that one of the norm $\|\cdot\|_1$ or $\|\cdot\|_2$ is fixed and a set of centers of clusters $y = (y^1, \ldots, y^k)$ is found by the k-means algorithm starting from a point $x = (x^1, \ldots, x^k)$. Let C_k be the cluster function with respect to either $\|\cdot\|_1$ or $\|\cdot\|_2$ (this does not depend on the fixed norm). Applying the Discrete Gradient algorithm with the initial point y for minimization of C_k we can find a point $z = (z^1, \ldots, z^k)$. We considered many initial points x and could not find an example where the inequality $C_k(x) > C_k(y) > C_k(z)$ does not hold, even if the norms, which was used for k-means and for the definition of C_k, did not coincide.

In the Table 11 we present results for both norms and only for one tolerance ε_h. In the Table 12 we present results only for $\|\cdot\|_1$ with different ε_h. The results obtained for different number of clusters $2 \leq k \leq 10$ are similar, so we present the results only for $k = 7$. We ran the programs with different ε_h in order to find the tolerance which allows reduction of the number of observations and keeps the approximation of the cluster-function reasonable.

In both tables the column *Size* describes the size of the dataset after ε_h-cleaning. The column *Distance* contains the information about the difference d_h between centres of clusters

Table 11: Choice of ε and clustering. Medium size datasets. 4 clusters.

Dataset	Norm	h	Size	Distances	Change of clusters
Liver (345)	1	0.9	133	0.73	0, -0.057, -0.079, 0.081
Liver (345)	2	0.9	78	0.61	0,075, -0.077, -0.032, 0.013
Diabetes (768)	1	0.6	290	1.46	-0.047, 0.075, -0.004, 0.008
Diabetes (768)	2	0.6	200	1.59	0.102, 0.042, -0.018, 0.138
Heart (297)	1	0.9	57	3.87	0, -0.027, 0.036, -0.010
Heart (297)	2	0.9	106	0.48	0, 0, 0, 0

Table 12: Choice of ε and clustering. Emergency dataset.

h	Size	Distance	Change of clusters
0	15 186	0	0, 0, 0, 0, 0, 0, 0
0.1	5989	1.09	0.009, -0.102, 0.009, -0.021, 0.009, 0.024, -0.09, -0.007
0.2	2740	1.00	-0.010, 0.007, 0.012, 0.011, -0.025, 0.008, 0.009
0.3	1487	1.07	0.075, -0.035, -0.009, 0.008, 0.158, -0.049, -0.153
0.4	938	1.53	-0.390, 0.150, 0.255, -0.016, 0.302, 0.098, -0.032
0.5	652	2.32	-0.289, 0.238, 0.165, 0.017, 0.340, 0.139, -0.066

with and without ε_h-cleaning: $d_h = \frac{1}{k}\sum_{i=1}^{k} \|\bar{x}^i - \bar{x}^i_h\|$. Here k is the number of clusters; $\bar{x}_i$ are the centres of clusters of the original dataset, obtained by the minimization of the cluster function, and $\bar{x}^i_h$ the centers obtained by the minimization of the generalized cluster functions corresponding to ε_h. The column *Change of clusters* contains the numbers $R^i_h,\ i = 1, \ldots, k$ that characterize the change of the size of each cluster. By definition

$$R^i_h = \frac{N_{or} - N^i_{clear}}{N^i_{or}},$$

where N^i_{or} is the number of points within the cluster i, that obtained by the minimization of the cluster function, and N^i_{clear} is the number of points within this cluster, obtained by the minimization of the generalized cluster functions. Table 11 contains also the columns $Dataset$, where the name of the dataset and its size (in brackets) are shown and the column $Norm$.

Table 12 demonstrates that even reducing the size of the Emergency dataset by more than 10 times we get satisfactory results. The obtained results also show that even very rough approximation of the cluster function leads to satisfactory results.

Comparing Table 11 and Table 12 we can suggest that the percentage of points that can be removed without destroying the structure of datasets heavily depends on the size of the dataset under consideration: for a larger dataset this percentage is bigger. Indeed, it follows from the following simple observation: if two datasets are located in the same volume then points from a larger dataset are closer to each other. This conclusion allows us to hope that ε_h cleaning procedure can be successfully used for clustering very large datasets.

The following conclusion is also of interest. Two different approaches to choosing ε: the first approach is based on the minimal distance between the points and the second one controls the relative error left almost the same number of points after "cleaning".

2.12 Geometry of finite sets

Clusters can describe the structure of finite sets of points. It is important also to describe some different characteristics of finite sets, which can help in the study of these sets and also classification problems related to them. We give an example of such characteristics (see [79] for details).

Example 2.1 Let A be a finite set of points. We describe this set by a collection of hyperplanes.

Consider vectors $l_1, \ldots, l_k$ with $\|l_i\| = 1$ and numbers b_i $(i = 1, \ldots, k)$. Let $H_i = \{x : [l_i, x] = b_i\}$ and $H = \cup_i H_i$. Then the distance between the set H_i and a point a^q is $d(a^q, H_i) = |[l_i, a^q] - b_i|$ and the distance between the set H and a^q is

$$d(a^q, H) = \min_i |[l_i, a^q] - b_i|.$$

The deviation of X from A is

$$\sum_{q \in Q} d(a^q, H) \equiv \sum_{q \in Q} \min_i |[l_i, a^q] - b_i|$$

Consider the function

$$L_k((l_1, b_1), \ldots, (l_k, b_k)) = \sum_{q \in Q} \min_i |[l_i, a^q] - b_i|.$$

A solution of the following constrained *min-sum-min* problem

$$\min_{(l_1,b_1) \in \mathbb{R}^{n+1}, \ldots, (l_k,b_k) \in \mathbb{R}^{n+1}} \sum_{q \in Q} \min_i |[l_i, a^q] - b_i|$$

subject to

$$\|l_1\| = 1, \ldots, \|l_k\| = 1$$

describes the *skeleton* of the set A, which is formed by k hyperplanes.

It is easy to give some examples of finite sets, for which skeletons are a better description of the set than clusters. Skeletons can also find some applications in classification.

3 Supervised classification via clustering

3.1 Introduction

The aim of supervised data classification is to establish rules for the classification of some observations assuming that the classes of data are known. To find these rules, an investigator can use known training subsets of the given classes. The construction of a classification procedure may also be a pattern recognition procedure, a discrimination procedure or supervised learning procedure. Those problems arise in a wide range of human activity.

There are many methods for data classification, which are based on quite different approaches (statistics, neural networks, methods of information theory etc). Excellent review of these

approaches, including their computational investigation and comparison, can be found in [67]. Statistical approaches to classification are described in [64].

One of the most promising approaches to data classification is based on methods of mathematical optimization. For supervised classification, where we have a database, which consists of at least two classes and there is a training set for each class, there are two different ways for the application of optimization. The first, which we shall call *outer*, is based on the separation of the given training sets by means of a certain (not necessary linear) function. The outer approach is currently the most popular. (See, for example, [28, 29, 63], where problems of quadratic and bilinear programming are used for classification and then linear programming techniques are applied for the solution of these problems.) The second (*inner*) approach consists of describing clusters for the given training sets. The data vectors are assigned to the closest cluster and correspondingly to the set, which contains this cluster. The description of this approach can be found in the recent paper [18]. Numerical experiments demonstrate that for supervised classification of databases of a small to medium size, the inner approach gives a more precise description of databases than the outer approach. We examine the inner approach in this section. For the implementation of this approach one needs to solve a complex problem of nonsmooth and nonconvex unconstrained optimization, either local or global. In spite of the nonsmoothness and nonconvexity of the objective function, local methods are much simpler and more applicable, than global ones.

On the other hand global methods give more precise descriptions of clusters. A deterministic method for solving global optimization problems (the cutting angle method) has recently been developed [6, 19, 20, 23]. Some modifications of the cutting angle method and its combination with a local search ([21]) were successfully applied to classification. Our numerical experiments with real-world databases of small to medium size show that the inner approach to the supervised classification problem based on optimization techniques gives results close to the best known ones obtained by different methods. (see Section 3.8).

3.2 The inner approach to classification

Assume that we have a dataset consisting of l classes $A_1, \ldots, A_l$. Assume that the class A_j consists of k_j clusters. We can use the minimization of the cluster function C_{k_j} in order to find centers of these clusters and then use these clusters for classification. From the first view this approach is not suitable. Indeed, actually we reduce the more simple problem of supervised classification to the series of more complicated problem of unsupervised classification. However, the presence of known classes substantially facilitates the search for clusters.

The cluster function C_k depends on $n \times k$ variables, where n is the number of attributes (features) and k is the number of clusters. If classes are known, we can apply a certain feature selection procedure (see Subsection 3.3) for diminishing the number of features, hence we can use only $n_1 < n$ features. Second, we can determine the centers of clusters step by step. This means that we consider a series of k problems of the dimension n_1 instead of one problem of the dimension $n_1 \times k$. The solution of this series is much easier then solving one high-dimensional problem. We now explain why step by step procedure can be used for supervised classification.

Let A consists of two classes, A_1 and A_2. Assuming that the class A_j $(j = 1, 2)$ consists of only

one cluster we can calculate its centre by solving the following convex programming problem:

$$\text{minimize } C_1(x) = \sum_{a \in A_j} \|x - a\| \quad \text{subject to } x \in \mathbb{R}^n. \tag{3.1}$$

The centre of A_1 reflects the influence of all clusters from this set. Having the centre of A_2, we can define misclassified points of A_1 as points that are closer to the centre of A_2 than to the centre of A_1. Removing all misclassified points and solving problem (3.1) for $j = 1$ again we can find a more precise centre x^1 of the set A_1. We can consider x^1 as the centre of the first (main) cluster of the set A_1. Then we can look for the centre of the second cluster with the known centre of the first cluster etc. The similar idea can be used for feature selection. If we have a large-scale dataset we can use also complexity reduction procedure (see Subsections 2.10 and 2.11) in order to reduce the number of records.

3.3 Feature selection algorithm

In this section we describe a feature selection algorithm. A more detailed description of this algorithm can be found in [17].

The purpose of a feature selection procedure is to find as small a set as possible of informative features of the dataset under consideration, which describe this set from the point of view of classification. Some statistical methods are usually used for feature selection (for example, principal component analysis, see [69] and references therein). We accept a different approach to a feature selection procedure that is based on different understanding of the notion of *informative* feature. We assume that informativeness is not an individual property of feature, it depends on the classification structure of a dataset under consideration (see [17] for details). Roughly speaking a set of informative features should help to distinguish classes. It is better to consider not individual informative features, but subsets of informative features. The set of all features can contain different subsets of informative features. The simplest example: if values of x_1 are proportional to values of x_2, we can consider either x_1 as informative and x_2 as superfluous or x_2 as informative and x_1 as superfluous.

We use the centres of classes for the feature selection. These centres, which can be found by solving the convex programming problem (3.1), can give some information about the structure of classes. Then using some rules we can step-by-step remove some features while the structure of the classes begins to be changed. This idea leads to an algorithm for the solution of the feature selection problem.

We consider a database which contains m nonempty finite sets $A_j \subset \mathbb{R}^n,\ j = 1, \ldots, m$ with $m \geq 2$. Let

$$N_j = \{1, \ldots, |A_j|\},\ j = 1, \ldots, m,$$

where $|A|$ denotes the cardinality of a finite set A. First assume that $m = 2$.

Let $\varepsilon > 0$ be some tolerance and $T_j \in \{1, 2, \ldots\},\ j = 1, 2, 3$ be the thresholds.

Algorithm 3.1 Feature selection

Step 1. *Initialization.* Set $k = 0,\ I_k = \{1, \ldots, n\}$.

Step 2. *Determine centers of clusters by assuming that the sets $A_j,\ j = 1, 2$ contain a unique cluster.* Compute the centers of clusters by solving the following problems of convex programming:

$$\text{minimize} \quad \sum_{i \in N_j} \|x^j - a^{ij}\|_p \tag{3.2}$$

subject to $x^j \in \mathbb{R}^n,\ j = 1, 2$. Here $\|\cdot\|_p$ is defined by

$$\|x\|_p = \left(\sum_{l \in I_k} |x_l|^p \right)^{1/p}.$$

Step 3. *Find points of the set $A_j,\ j = 1, 2$, which are closer to the cluster center of the other set.*

Let $x^j_*,\ j = 1, 2$ be solutions to (3.2). Compute the sets:

$$N_1^k = \{i \in N_1 : \quad \|x^2_* - a^{i1}\|_p \le \|x^1_* - a^{i1}\|_p\},$$

$$N_2^k = \{i \in N_2 : \quad \|x^1_* - a^{i2}\|_p \le \|x^2_* - a^{i2}\|_p\}.$$

Set

$$N_3^k = N_1^k \bigcup N_2^k.$$

If $k = 0$ then go to Step 5, otherwise go to Step 4.

Step 4. Calculate

$$L_j^{max} = \max\{|N_j^t| : t = 0, \ldots, k\},\ j = 1, 2,$$

$$L_3^{max} = \max\{|N_1^t| + |N_2^t| : t = 0, \ldots, k\}.$$

If

$$\max\{L_j^{max} - T_j :\ j = 1, 2, 3\} > 0$$

then I_{k-1} is a subset of most informative attributes and the algorithm terminates. Otherwise go to Step 5.

Step 5. *To determine the closest coordinates.* Calculate

$$d_0 = \min\{|(x^1_*)_l - (x^2_*)_l| : l \in I_k\},$$

and define the following set:

$$R_k = \{l \in I_k : \quad |(x^1_*)_l - (x^2_*)_l| \le d_0 + \varepsilon\}.$$

Step 6. Construct the set:

$$I_{k+1} = I_k \setminus R_k.$$

If $I_{k+1} = \emptyset$ then I_k is the subset of most informative attributes. If $|I_{k+1}| = 1$ then I_{k+1} is the subset of most informative attributes. Then the algorithm terminates, otherwise set $k = k + 1$ and go to Step 2.

Remark 3.1 Different norms can lead to different sets of informative features.

Remark 3.2 Assume that the number m of classes is greater than 2. Then the version of Algorithm 3.1 can be considered, where Steps 3, 4 and 5 are replaced by Steps $3', 4'$ and $5'$, respectively.

Step $3'$. *To find points of a set $A_j,\ j = 1, \ldots, m$, which are closer to the cluster centers of other sets.*

Let $x_*^s,\ s = 1, \ldots, m$ be solutions to the problem (3.2). Compute the sets:

$$N_j^k = \{i \in N_j : \quad \min_{s=1,\ldots,m, s\neq j} \|x_*^s - a^{ij}\|_p \leq \|x_*^j - a^{ij}\|_p\},\ j = 1, \ldots, m.$$

Set

$$N_{m+1}^k = \bigcup_{j=1}^{m} N_j^k.$$

If $k = 0$ then go to Step $5'$, otherwise go to Step $4'$.

Step $4'$. Calculate

$$L_j^{max} = \max\{|N_j^t| : t = 0, \ldots, k\},\ j = 1, \ldots, m,$$

$$L_{m+1}^{max} = \max\{\sum_{j=1}^{m} |N_j^t| : t = 0, \ldots, k\}.$$

If

$$\max\{L_j^{max} - T_j :\ j = 1, \ldots, m+1\} > 0$$

then I_{k-1} is a subset of most informative attributes and the algorithm terminates. Otherwise go to Step $5'$.

Step $5'$. *To determine the closest coordinates.* Calculate

$$d_l = \max\{|(x_*^i)_l - (x_*^t)_l| : i, t = 1, \ldots, m\},$$

$$d_0 = \min\{d_l : l \in I_k\}$$

and define the following set:

$$R_k = \{l \in I_k \mid d_l \leq d_0 + \varepsilon\}.$$

Remark 3.3 Note that the subset of most informative attributes calculated by the Algorithm 3.1 depends on the vector of thresholds

$$T = (T_1, T_2, \ldots, T_{m+1}).$$

The choice of this vector depends on the aim of the researches.

3.4 Feature selection by clustering

Sometimes[2] we have to use very small datasets. This happens when the cost of each experiment for obtaining a new data point is very expensive. For many such datasets feature selection is the most important procedure. It is impossible to use statistical methods for feature selection for datasets of small size. At the same time for a small dataset we can minimize cluster function with $k > 1$. Using this procedure we can obtain a much more effective feature selection procedure than in the case $k = 1$.

The conceptual scheme of the the proposed method has the following form:

1. Find the centres for each class A_j, and the clusters C_j^l associated to these centres;
2. Determine the order of importance of the features;
3. Eliminate the least important feature, and calculate the new centres;
4. Calculate the clusters $(C')_j^l$ associated to the new centres.
5. If $C_j^l = (C')_j^l$, then go to 3 otherwise stop.

The centres for each class can be carried out by the minimization of the cluster function. For determining the order of importance of the features we can use the same approach as in Algorithm 3.1, which is based on the comparison of the difference between each coordinate of the centres. Then the least informative feature is eliminated and the new dataset is studied. The new centres and the new clusters are calculated. If these clusters are different from the previous ones, then the feature that were just eliminated is informative (its elimination leads to a different result), and the process is stopped. Otherwise, the feature is not informative and it can be eliminated. The elimination process starts once again.

It is possible to calculate the centres according to different norms. The choice of the norm influences the results: different norms lead to different orders for the features. When the calculations are made for several norms ($\|\cdot\|_1$ and $\|\cdot\|_2$ are the most common), it can happen that two features have a different order relation depending on the norm. In this case, these features are considered to be *equivalent*. If on the contrary, the features have the same order relation, then it is considered that one features is clearly more informative than the other.

Thus it is possible to create some groups of equivalent features. Each group represents a level of information, and they can be classified in the same order relationship, independently from the norm.

The interest in the definition of the levels of information is that they define groups of features that can be considered as having equivalent information, and these groups are ordered independently from the method of finding the centres. Numerical experiments confirm that the proposed method can be successfully applied for feature selection.

3.5 Step by step procedure for finding centers of clusters

The refining procedure, that is based on the presence of known classes, allows one to find clusters of the given dataset step-by-step. Let A_j be one of the given classes of the dataset A.

[2]The results from this subsection have been obtained by J. Ugon

Assuming that this set consists of the one cluster we can find the center of this cluster (that is, the centre of the set A) by solving problem (3.1). Using the refining procedure, we can remove all misclassified points and then solve problem (3.1) only for the rest of the points. The solution x^1 of this problem can be considered as the centre of the first cluster.

In order to find a center of the second cluster we solve the following problem:

$$\text{minimize} \quad f_2(x) = \sum_{i=1}^{r} \min\{\|x^1 - a^i\|, \|x - a^i\|\} \quad \text{subject to} \quad x \in \mathbb{R}^n, \tag{3.3}$$

where r stands for the number of records in the class A_j.

Assume that we have already calculated the centres $x^1, \ldots, x^{t-1}$ of $t-1$ clusters, then the center x^t of t-th cluster is defined as a solution to the following problem:

$$\text{minimize} \quad f_t(x) \quad \text{subject to } x \in \mathbb{R}^n, \tag{3.4}$$

where

$$f_t(x) = \sum_{i=1}^{r} \min\{\|x^1 - a^i\|, \ldots, \|x^{t-1} - a^i\|, \|x - a^i\|\}. \tag{3.5}$$

The number of variables in (3.4), which is n, is substantially less than that in (2.8). Note that the minimization of cluster function requires much more time than the solution of the series of problems (3.4).

Let

$$c_i^t = \min\{\|x^1 - a^i\|, \ldots, \|x^{t-1} - a^i\|\}, \ t \geq 2.$$

Then we can present the function f_t defined by (3.5), in the following form:

$$f_t(x) = \sum_{i=1}^{r} \min\{c_i^t, \|x - a^i\|\}, \ t \geq 2.$$

Thus we get the following problem of global optimization:

$$\text{minimize} \quad f_t(x) = \sum_{i=1}^{r} \min\{c_i^t, \|x - a^i\|\} \quad \text{subject to } x \in \mathbb{R}^n. \tag{3.6}$$

It is clear that

$$f_t(x) \leq f_{t-1}(x) \leq \ldots \leq f_1(x)$$

for all $x \in \mathbb{R}^n$. Let x_t^* be a solution to the problem (3.6). Then we have

$$0 \leq f_t(x_t^*) \leq f_{t-1}(x_{t-1}^*) \leq \ldots \leq f_1(x_1^*).$$

Thus we get a decreasing lower bounded sequence $\{f_t(x_t^*)\}$. Then this sequence converges and we obtain the following stopping criterion: if at t-th iteration

$$f_{t-1}(x_{t-1}^*) - f_t(x_t^*) \leq \varepsilon$$

where $\varepsilon > 0$ is some tolerance, then the further solving of the problem (3.6) will not improve the description of the given class A_j.

Assume without loss of generality, that $c_i^t > 0$ for all $i = 1, \ldots, r$. Indeed if $c_{i_0}^t = 0$ for some $i_0 \in \{1, \ldots, r\}$ then

$$f_t(x) = \sum_{i=1}^{r} \min\{c_i^t, \|x - a^i\|\} = \sum_{i=1, i \neq i_0}^{r} \min\{c_i^t, \|x - a^i\|\}$$

for all $x \in \mathbb{R}^n$ and so the point a^{i_0} can be deleted. Then

$$\bar{c}^t = \min\{c_i^t,\ i = 1, \ldots, r\} > 0.$$

From the definition of the function f_t we immediately obtain that

$$f_t(x) \leq \sum_{i=1}^{r} c_i^t \tag{3.7}$$

for all $x \in \mathbb{R}^n$. The following assertion holds (see [18]):

Proposition 3.1 *We have: $f_t(x) \geq \bar{c}^t$ for all $x \in \mathbb{R}^n$.*

3.6 The main algorithm for classification

In this section we give a description of the main algorithm for the solution of classification problems.

In this section p-norms with either $p = 1$ or $p = 2$ will be used again. We consider a database which contains 2 classes: A_1 and A_2. Let

$$N_1 = \{1, \ldots, |A_1|\}, \qquad N_2 = \{|A_1| + 1, \ldots, |A_1| + |A_2|\}.$$

We assume that the feature selection algorithm has been applied and a small subset of most informative attributes has been calculated. Let $\varepsilon > 0$ be a tolerance.

Algorithm 3.2 Classification algorithm

Step 1. *Initialization.* Determination of the centers of clusters, by assuming that sets A_1 and A_2 contain a unique cluster.

Compute the centers of clusters solving the following problems of convex optimization:

$$\text{minimize} \quad \sum_{i \in N_1} \|x^1 - a^i\|, \tag{3.8}$$

$$\text{minimize} \quad \sum_{i \in N_2} \|x^2 - a^i\| \tag{3.9}$$

subject to $x^j \in \mathbb{R}^n,\ j = 1, 2$. Set $k = 1$. Let x_{1k}^* and x_{2k}^* be solutions to the problems (3.8) and (3.9) and let f_{1k}^* and f_{2k}^* be the values of these problems, respectively.

Step 2. *Find the sets of points "misclassified" by the current clusters.* Compute the sets:

$$N_{1k}^* = \{i \in N_1 : \quad \min_{t=1,\ldots,k} \|x_{2t}^* - a^i\| \leq \min_{t=1,\ldots,k} \|x_{1t}^* - a^i\|\},$$

$$N_{2k}^* = \{i \in N_2 : \quad \min_{t=1,\ldots,k} \|x_{1t}^* - a^i\| \leq \min_{t=1,\ldots,k} \|x_{2t}^* - a^i\|\}.$$

Step 3. Compute the following sets:

$$K_1 = \{i \in N_1 \setminus N_{1k}^* : \quad \|x_{1k}^* - a^i\| \leq \min_{t=1,\ldots,k-1} \|x_{1t}^* - a^i\|\},$$

$$K_2 = \{i \in N_2 \setminus N_{2k}^* : \quad \|x_{2k}^* - a^i\| \leq \min_{t=1,\ldots,k-1} \|x_{2t}^* - a^i\|\}.$$

Step 4. *Refine the center of the cluster by using only vectors, which are closer to the center of this cluster.*

Solve the following convex programming problems:

$$\text{minimize} \quad \sum_{i \in K_1} \|x^1 - a^i\|, \tag{3.10}$$

$$\text{minimize} \quad \sum_{i \in K_2} \|x^2 - a^i\| \tag{3.11}$$

subject to $x^j \in \mathbb{R}^n,\ j = 1, 2.$

Let x^{01} and x^{02} be the solutions of the problems (3.10) and (3.11), respectively. Set $x_{1k}^* = x^{01}$ and $x_{2k}^* = x^{02}$.

Step 5. *Determine the next cluster.*

Solve the following optimization problems:

$$\text{minimize} \quad \sum_{i \in N_1} \min\{\|x^1 - a^i\|, \|x_{11}^* - a^i\|, \ldots, \|x_{1k}^* - a^i\|\}, \tag{3.12}$$

$$\text{minimize} \quad \sum_{i \in N_2} \min\{\|x^2 - a^i\|, \|x_{21}^* - a^i\|, \ldots, \|x_{2k}^* - a^i\|\} \tag{3.13}$$

subject to $x^j \in \mathbb{R}^n,\ j = 1, 2.$

Step 6. Let x^{11} and x^{12} be the solutions, and $f_{1,k+1}$ and $f_{2,k+1}$ be the values of the problems (3.12) and (3.13), respectively. Set $x_{1,k+1}^* = x^{11}$ and $x_{2,k+1}^* = x^{12}$.

Step 7. *Checking the stopping criterion.*

If

$$\max\left\{\frac{|f_{1,k+1} - f_{1k}|}{f_{11}}, \frac{|f_{2,k+1} - f_{2k}|}{f_{21}}\right\} < \varepsilon$$

then the algorithm terminates. Otherwise set $k = k + 1$ and go to Step 2.

Remark 3.4 In order to apply this algorithm to the investigation of concrete datasets we need to solve the minimization problem (3.6), since both (3.12) and (3.13) have a form (3.6). There are two modification of the algorithm, corresponding to local optimization and global optimization of (3.6), respectively. Then various methods for numerical optimization lead to various versions of this algorithm. We examine two versions based on different methods of local optimization and one version based on global optimization.

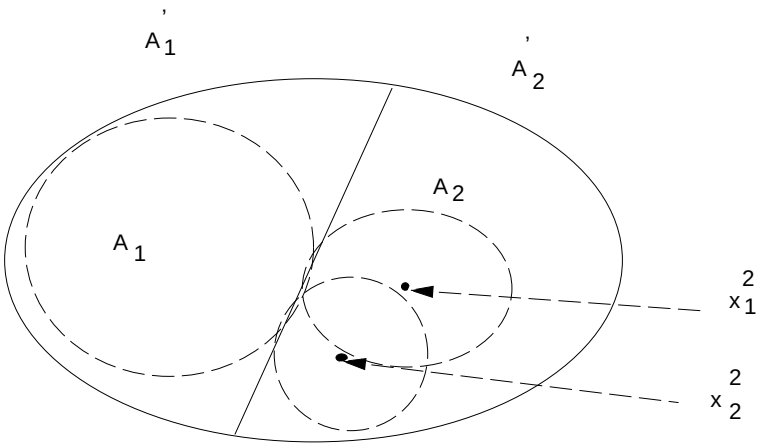

Figure 4: Classification via clustering

3.7 Implementation of algorithms

The proposed algorithms have been testing on real-world datasets. The Australian credit dataset, the Wisconsin breast cancer dataset, the diabetes dataset, the heart disease dataset and the liver-disorder dataset have been used in numerical experiments. The description of these datasets can be found in Appendix.

We tested three algorithms.

- Algorithm 1 is the version of the main classification Algorithm 3.2 with the local optimization in Step 5 based on a continuous approximations to the Clarke subdifferential (see Section 4);
- Algorithm 2 is the version of Algorithm 3.2 with the local optimization in Step 5 based on a continuous approximations to the Demyanov-Rubinov quasidifferential (see Section 4);
- Algorithm 3 is the version of Algorithm 3.2 with the global optimization in Step 5. Algorithm 4.1 (See Section 4) is used for global optimization. A local search is accomplished by a method based on the continuous approximations to the Demyanov-Rubinov quasidifferential. The cutting angle method for global optimization (see Section 4) is used in order to leave a stationary point that is found by a local search.

Remark 3.5 1) The maximum number of the discrete gradients in Algorithms 1,2 and 3 is 20.

2) Stopping criterion from Step 7 of Algorithm 3.2 with $\varepsilon = 10^{-2}$ is used for all Algorithms.

Remark 3.6 The number of iterations evaluated by the cutting angle method in Algorithm 3 is restricted. The cutting angle method evaluates at most 70 iterations for all cases.

Thus if the cutting angle method is not able to leave a stationary point for a given number of iterations, this point is considered as a surrogate to the global minimum. To make the

work of the Algorithm 3 more precise we need more powerful hardware and new more powerful modifications of the cutting angle method and its combinations with the local search. We also can use different techniques for global optimization. Nevertheless, even with mentioned restrictions, the Algorithm 3 leads to better results than algorithms, based on local optimization.

First we apply feature selection Algorithm 3.1 which calculates the subset of informative attributes. Results of numerical experiments show that this algorithm substantially reduces the number of attributes. So we use 3 attributes in the Australian credit dataset, the Wisconsin breast cancer dataset and the diabetes dataset, 11 attributes in the heart disease dataset and 6 attributes in the liver-disorder dataset for solving classification problem.

In numerical experiments we use p-norms with $p = 1$ and $p = 2$ and present the best results.

For comparison of the results of numerical experiments we chose 23 algorithms of classification from [67] and results of numerical experiments using these algorithms presented in Chapter 9 of this book. These are statistical, neural network and machine learning algorithms (see [67] for details). In the tables below we present only the best results obtained by these algorithms.

We also use results obtained by Hybrid misclassification minimization (HMM) [34], Parametric misclassification minimization (PMM) [62] and Robust linear programming (RLP) [24] algorithms. Finally we use results from [27] obtained by support vector machines algorithms SVM $\|\cdot\|_1$, SVM $\|\cdot\|_\infty$ and SVM $\|\cdot\|_2^2$ with 1-norm, ∞-norm and 2-norm, respectively.

For different databases results of different algorithms are available for comparison (for example, we cannot compare our results with HMM, PMM and RLP for all datasets). In order to provide comparison we consider different fold cross validations for the same dataset. For the heart disease dataset with 9-fold cross validation we consider a comparison of average cost, since only this parameter is available (for the definition of average cost see, for example, [67], Chapter 2).

3.8 Results of numerical experiments

In this section we present results of the numerical experiments. We use the following notation for the description of their results:

- n_f is the number of the objective function evaluations;
- t_1 is the average CPU time in the training phase for one cycle of cross-validation (in seconds);
- t_2 is the average CPU time in the testing phase for one cycle of cross-validation (in seconds);
- e_1 is the error rate for the training data;
- e_2 is the error rate for the test data;
- ac_1 is the average cost for the training data;
- ac_2 is the average cost for the test data;

- N_c is the number of calculated clusters.

Additionally, in the tables below in the column *"norm"* we give a norm whereby the best result has been obtained.

The code has been written in Fortran 90 and the numerical experiments have been carried out on a PC Pentium-S with CPU 150 MHz.

Remark 3.7 It should be noted that for each dataset and one cycle of cross-validation we solve various optimization problems $2mN_c$ times where m is the number of classes and N_c is the number of calculated clusters.

3.8.1 Australian credit database

This database has been studied before by J.R. Quinlan (see [73]). It contains 2 classes, 690 observations and 14 attributes where 8 of them are categorical. We used 10-fold cross validation. The results of numerical experiments are presented in Table 13.

Table 13: Results for Australian credit database

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	8574	7.51	0.06	0.145	0.144	2	2
Algorithm 2	11127	20.50	0.06	0.145	0.144	2	2
Algorithm 3	20820	87.90	0.06	0.130	0.128	2	3
Best result from [67]	-	-	-	0.132	0.131	-	-

The results presented in Table 13 show that Algorithm 3 gives best accuracy however it uses much more computational efforts than other algorithms. The accuracies of Algorithms 1 and 2 are the same, however Algorithm 1 used less CPU time.

3.8.2 Wisconsin Breast Cancer database

In the study of this dataset we used 10-fold cross validation. The results of numerical experiments are presented in Table 14.

Table 14: Results for Wisconsin Breast Cancer database

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	34234	27.10	0.06	0.025	0.032	2	3
Algorithm 2	107752	125.33	0.06	0.011	0.016	2	3
Algorithm 3	146310	235.41	0.06	0.012	0.014	2	3
HMM	-	-	-	0.021	0.026	-	
PMM	-	-	-	0.014	0.035	-	
RLP	-	-	-	0.023	0.028	-	

The results presented in Table 14 show that the accuracies of Algorithms 2 and 3 are almost the same, however Algorithm 2 uses significantly less computational efforts. The accuracy of Algorithm 2 two times less than that of Algorithm 1, however the accuracy of Algorithm 1 is still high enough. It should be noted that Algorithm 1 uses less CPU time and the objective function evaluations than Algorithms 2 and 3.

3.8.3 The diabetes database

We used 10 and 12-fold cross validation. The results of numerical experiments with 10 and 12-fold cross validation are presented in Tables 15 and 16, respectively.

Table 15: Results for the diabetes database with 10-fold cross validation

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	159091	133.94	0.06	0.179	0.221	1	9
Algorithm 2	344446	855.00	0.06	0.180	0.236	2	9
Algorithm 3	412322	1095.15	0.06	0.173	0.214	2	9
HMM	-	-	-	0.216	0.241	-	-
PMM	-	-	-	0.194	0.233	-	-
RLP	-	-	-	0.233	0.240	-	-
SVM $\|\cdot\|_1$	-	-	-	0.248	0.254	-	-
SVM $\|\cdot\|_\infty$	-	-	-	0.245	0.255	-	-
SVM $\|\cdot\|_2^2$	-	-	-	0.240	0.250	-	-

Table 16: Results for the diabetes database with 12-fold cross validation

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	146824	125.96	0.06	0.223	0.259	1	9
Algorithm 2	246461	329.85	0.06	0.180	0.214	2	9
Algorithm 3	289455	452.77	0.06	0.174	0.199	2	9
Best result from [67]	-	-	-	0.219	0.223	-	-

The results presented in Tables 15 and 16 allow us to conclude that the accuracy of Algorithm 3 was best for this dataset however this algorithm again uses much more CPU time. The accuracies of Algorithms 1 and 2 are high enough.

3.8.4 The heart disease database

In order to provide a comparison we present average cost in Table 17. We used 9- and 10-fold cross validation and took 270 instances with 9-fold cross validation. The results of numerical experiments with 9- and 10-fold cross-validation are presented in Tables 17 and 18, respectively.

The accuracy of Algorithm 3 is higher than one for other algorithms. On the same time

Table 17: Results for the heart disease database with 9-fold cross validation

Algorithm	n_f	t_1	t_2	ac_1	ac_1	norm	N_c
Algorithm 1	71378	36.09	0.06	0.438	0.455	1	2
Algorithm 2	92584	153.41	0.06	0.428	0.415	1	2
Algorithm 3	97624	183.10	0.06	0.412	0.374	1	2
Best result from [67]	-	-	-	0.351	0.374	-	-

Table 18: Results for the heart disease database with 10-fold cross validation

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	41936	22.61	0.06	0.173	0.159	1	2
Algorithm 2	97785	176.01	0.06	0.165	0.152	1	2
Algorithm 3	104025	198.11	0.06	0.161	0.148	1	2
HMM	-	-	-	0.125	0.172	-	-
PMM	-	-	-	0.086	0.178	-	-
RLP	-	-	-	0.155	0.165	-	-
SVM $\|\cdot\|_1$	-	-	-	0.147	0.154	-	-
SVM $\|\cdot\|_\infty$	-	-	-	0.142	0.175	-	-
SVM $\|\cdot\|_2^2$	-	-	-	0.153	0.241	-	-

it takes mush more CPU time. The accuracies of Algorithms 1 and 2 are high enough in comparison with methods based on other optimization methods.

3.8.5 The liver-disorder database

We used 10-fold cross validation. The results of numerical experiments are presented in Table 19.

The results presented in Table 19 show that the accuracy of Algorithm 3 is the best however it uses much more computational efforts than other algorithms.

3.9 Discussion of results of numerical experiments

The accuracy is one of the most important criteria for evaluation of classification methods. Usually cross validation is used to estimate the accuracy of the methods for databases of medium size. However cross validation can not be used for the precise estimation of the method. Indeed, as it follows from Tables 15 and 16 , the accuracy of algorithms for diabetes database depends not only on methods and database but also on the kind of cross-validation that was used. Thus, the error e_2 for Algorithm 3 is 0.214 for 10-fold cross-validation and 0.199 for 12-fold cross-validation, so Algorithm 3 gives more precise solution in terms of 12-fold cross-validation. At the same time e_2 for Algorithm 1 is 0.221 for 10-fold cross-validation and 0.259 for 12-fold cross-validation so Algorithm 1 gives more precise solution in terms of 10-fold cross-validation. Thus we cannot give an exact estimation of the accuracy of an

Table 19: Results for the liver-disorder database

Algorithm	n_f	t_1	t_2	e_1	e_2	norm	N_c
Algorithm 1	87215	47.19	0.06	0.255	0.309	2	6
Algorithm 2	81299	103.71	0.06	0.260	0.294	2	6
Algorithm 3	123127	142.26	0.06	0.254	0.274	2	6
HMM	-	-	-	0.278	0.334	-	-
PMM	-	-	-	0.251	0.316	-	-
RLP	-	-	-	0.310	0.331	-	-
SVM $\|\cdot\|_1$	-	-	-	0.322	0.360	-	-
SVM $\|\cdot\|_\infty$	-	-	-	0.313	0.354	-	-
SVM $\|\cdot\|_2^2$	-	-	-	0.398	0.390	-	-

algorithm by using cross-validation.

However, the tables presented demonstrate that Algorithm 3 outperforms Algorithms 1 and 2, all 23 algorithms from [67] and also algorithms from [34, 62, 27]. At the same time Algorithm 3 requires much more CPU time than Algorithms 1 and 2. We could not compare CPU time required for Algorithm 3 and algorithms from [67], however, we guess that Algorithm 3 is more time-consuming. Thus Algorithm 3 is beneficial if we need to get more precise results and we have no tight time frameworks.

3.10 Australian credit database

The accuracy of algorithms depends very strongly not only on the method, which is used for classification but also on the quality of database under consideration. We mention here a recent research [78], where the Australian credit database was investigated. Three different algorithms for classification were applied, which are based on completely different approaches: one of them (Algorithm LO) is a version of Algorithm 2 from the present paper, which is based on optimization. The second algorithm (Algorithm NN) was based on a neural network technique and the third one (Algorithm ML)was based on a machine learning technique. It turned out that Algorithm LO gives 102 misclassified points. Algorithm NN gives 100 misclassified points and all these points were from the subset of misclassified points, obtained by Algorithm LO. The ML Algorithm gave 74 misclassified points. The intersection of all three sets of misclassified points consists of 65 points. Because these three algorithms are based on completely different techniques we can conclude that these 65 points can be considered as questionable ones. The algorithms cannot achieve a high level accuracy because of the presence of these questionable points.

The structure of clusters, which were found by Algorithm LO, confirms this conclusion. The feature selection algorithms demonstrates that we can use a subset of three informative features $8, 9, 14$ for classification. Two of these features, namely 8 and 9 are categorical, so we need to be careful with optimization algorithms. It was found that Australian credit database consists of 5 clusters. Two of these clusters contains mainly points from the first class, two other contains mainly points from the second class and the fifth cluster consists of mixture of points from both classes, which can not be separated (see Table 20. It is appeared that

the majority of misclassified points are located in the fifth cluster. Since the the most part of misclassified points was discovered by algorithms of very different nature, we got a confirmation that the optimization algorithm for finding clusters works well in spite of presence of categorical features.

After removing 65 questionable points, Algorithm LO (a version of Algorithm 2) classified points with the accuracy 94.5%. Note that the ML algorithm gives even better solution.

Table 20: Clusters for Australian credit dataset

	cluster I	cluster II	cluster III	cluster IV	cluster V
class I	241	63	21	2	56
class II	21	2	150	66	68

4 Numerical methods

It is assumed that the reader of this section is familiar with such notions of nonsmooth analysis as Clarke subdifferentials and quasidifferential (see, for example, [37]) and semismooth functions [68].

Optimization problems, which we use for clustering and classification have objective functions of the form

$$F_k(x^1, \dots, x^k) = \gamma \sum_{a \in A} \min_{i=1,\dots,k} \psi_i(x^i, a), \tag{4.1}$$

where the function $x \mapsto \psi_i(x, a)$ is convex for each i and a. We now give some examples of functions of the form (4.1).

1) Cluster function

$$C_k(x^1, \dots, x^k) = \frac{1}{N} \sum_{a \in A} \min_{i=1,\dots,k} \|x_i - a\|$$

has the form (4.1) with $\gamma = 1/N$ and $\psi_i(x, a) = \|x - a\|$ for all i;

2) The generalized cluster function

$$\tilde{C}_k(x^1, \dots, x^k) = \frac{1}{N} \sum_{a \in A} \min_{i=1,\dots,k} m_a \|x^i - a\|$$

has the form (4.1) with $\gamma = 1/n$ and $\psi_i(x, a) = m_a \|x - a\|$.

3) The function

$$f(x) = \sum_{a \in A} \min\{c_a, \varphi(x, a)\}$$

also has the form (4.1) with $\gamma = 1$, $k = 2$, $\psi_1(x, a) = c_a$, $\psi_2(x, a) = \varphi(x, a)$

The function L_k, which was used for the definition of skeletons of finite sets of points (see Subsection 2.12) also can serve an an example of function of the form (4.1).

Proposition 4.1 *The function F_k defined by (4.1) is DC (the difference of convex functions).*

Indeed, we can present F_k in the form (see for example [37],p.108):

$$F_k(x) = \gamma(f_1(x) - f_2(x)), \qquad x = (x_1, \ldots, x_k),$$

where

$$f_1(x) = \sum_{a \in A} \sum_{i=1}^{k} \varphi_i(x^i, a), \quad f_2(x) = \sum_{a \in A} \max_{i=1,\ldots,k} \sum_{j \neq i} \varphi_j(x^j, a).$$

Both f_1 and f_2 are convex functions.

Corollary 4.1 *The function F_k is quasidifferentiable, locally Lipschitz and semi-smooth.*

4.1 Methods for local optimization

In this section we discuss the Discrete Gradient method which can be used for local minimization of a function F_k defined by (4.1). Numerical experiments confirm that this method can escape from stationary points, which are not local minimizers and even from some shallow local minimizers so this method is very useful for the search for "deep" local minima. For the sake of simplicity we shall discuss only the minimization of function of the form

$$f(x) = \sum_{i=1}^{r} \min\{c_i, \varphi_i(x)\}, \tag{4.2}$$

where $\varphi_i(x) = \|x - a^i\|$. However all discussions below can be easily transform to the minimization of an arbitrary function F_k.

Due to Proposition 4.1 the minimization of f can be done by methods of nonsmooth optimization based on continuous approximations either to the Clarke subdifferential or quasidifferential. We apply both of them. Let

$$\eta_i(x) = \min\{c_i, \varphi_i(x)\}, \ i = 1, \ldots, r.$$

The Clarke subdifferential $\partial\eta_i(x)$ of the function η_i at a point $x \in \mathbb{R}^n$ has the following form:

$$\partial\eta_i(x) = \begin{cases} \{0\} & \text{if } c_i < \varphi_i(x), \\ \text{co}\,\{0, \partial\varphi_i(x)\} & \text{if } c_i = \varphi_i(x), \\ \partial\varphi_i(x) & \text{if } c_i > \varphi_i(x). \end{cases}$$

Here $\partial\varphi_i(x)$ is the subdifferential of the function φ_i at the point x in the sense of convex analysis.

It follows from well-known properties of the Clarke subdifferential that

$$\partial f(x) \subset \sum_{i=1}^{r} \partial\eta_i(x). \tag{4.3}$$

The function f is not regular in the sense of Clarke and consequently we cannot replace the inclusion for the equality in (4.3). Moreover the set on the right hand in (4.3) can be

much larger than the set on the left side. So we need approaches for the estimation of the subdifferential $\partial f(x)$, which are different from (4.3). We shall use an approach based on the notion of discrete gradients for this purpose.

The discrete gradient, which allows the construction of continuous approximations to the Clarke subdifferential was introduced and studied in [9, 11, 10]. A terminating algorithm for the computation of a descent direction of an objective function by means of discrete gradients has been described in [12]. We shall use versions of the discrete gradient method described in [12, 13] for the minimization of cluster function (see problem (2.8)) and also for solving problems (3.12) and (3.13), which appear in Algorithm 3.2.

Another approach to the minimization of the nonsmooth function f defined by (4.2) is based on continuous approximation of its quasidifferential. The function f can be presented as the difference of two convex functions:

$$f(x) = f_1(x) - f_2(x)$$

where

$$f_1(x) = \sum_{i=1}^{r} \varphi_i(x) + \sum_{i=1}^{r} c_i, \quad f_2(x) = \sum_{i=1}^{r} \max\{c_i, \varphi_i(x)\}.$$

Let

$$\theta_i(x) = \max\{c_i, \varphi_i(x)\}, \ i = 1, \ldots, r.$$

Then the function f is quasidifferentiable and its quasidifferential has the following form (see [37]):

$$Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$$

where

$$\underline{\partial} f(x) = \partial f_1(x) = \sum_{i=1}^{r} \partial \varphi_i(x),$$

$$\overline{\partial} f(x) = \partial f_2(x) = \sum_{i=1}^{r} \partial \theta_i(x).$$

Here $\partial \varphi_i(x)$ $(\partial \theta_i(x))$ is a subdifferential of the function φ_i (θ_i) at a point x in the sense of convex analysis. The subdifferential $\partial \theta_i(x)$ has the following form:

$$\partial \theta_i(x) = \begin{cases} \{0\} & \text{if } c_i > \varphi_i(x), \\ \operatorname{co}\{0, \partial \varphi_i(x)\} & \text{if } c_i = \varphi_i(x), \\ \partial \varphi_i(x) & \text{if } c_i < \varphi_i(x). \end{cases}$$

The continuous approximations to the quasidifferential and methods for their construction have been examined in [13, 10]. Methods for the minimization of functions presented as the difference of two convex functions, which are based on continuous approximations to the quasidifferential have been studied in [14]. We shall apply these methods for solving problems (3.12) and (3.13) in Algorithm 3.2.

4.2 Global optimization

In this subsection we will consider a method of search for the global minimum in (3.6), which is a combination of the cutting angle method [6] and a local search. We exploit a version of the cutting angle method which was proposed and studied in [19].

The cutting angle method can be applied to the global minimization of a Lipschitz function over the unit simplex

$$S = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^{n} x_i = 1\}.$$

Here $\mathbb{R}^n_+$ is the cone of vectors with nonnegative coordinates. Let f be a Lipschitz function defined on the simplex S with Lipschitz constant L. First we need to find a constant $c \geq c'$, where

$$c' = 2L - \min_{x \in S} f(x). \tag{4.4}$$

The cutting angle method can be applied for the minimization of the function $g(x) = f(x) + c$ over S. (See [76] for details.) Let $l = (l_1, \ldots, l_n) \in \mathbb{R}^n_+$. Consider the set of indices $I(l) = \{i : l_i > 0\}$. The vector l defines the so-called min-type function $x \mapsto \min_{i \in I(l)} l_i x_i$.

Subsequently we will use the unit vectors $e^m = (0, \ldots, 0, 1, 0, \ldots, 0)$ with $I(e^m) = \{m\}$.

The cutting angle method consists of two parts:

1) The choice of a certain strictly positive vectors $l^{n+1}, \ldots, l^j$, $j \geq n+1$. Then for each $j > n$ the sequence $l^1, \ldots, l^n, l^{n+1}, \ldots, l^j$ is considered, where $l^k = f(e^k)$ for $k = 1, \ldots, n$.

2) The solution of an auxiliary problem

$$\text{minimize} \quad h_j(x) \quad \text{subject to} \quad x \in S,$$

where

$$h_j(x) = \max_{k \leq j} \min_{i \in I(l^k)} l^k_i x_i. \tag{4.5}$$

The most difficult and time-consuming part of the cutting angle method is solving the auxiliary problem. A method for its solution was proposed in [19]. Some modifications of this method (and corresponding modifications of the cutting angle method) are discussed in details in [20]. We use these modifications for numerical experiments.

The cutting angle method is derivative free; only one value of the objective function is calculated at each iteration. Some modifications of this method require evaluation of a few values of the objective function at each iteration.

Consider now the following problem of unconstrained global optimization:

$$\text{minimize} \quad f(x) \quad \text{subject to } x \in \mathbb{R}^n. \tag{4.6}$$

Let x^* be a solution to (4.6). Assume that we know a vector of lower bounds $(a_1, \ldots, a_n)$ and a vector of upper bounds $(b_1, \ldots, b_n)$ of the unknown point x^*. Let $x'_i = x_i - a_i$ and $d = \sum_{i=1}^n (b_i - a_i)$. Then $x'_i \geq 0$ and $\sum_{i=1}^n x'_i \leq d$. Let $z_i = x'_i/d$, $i = 1, \ldots, n$ and $z_{n+1} = 1 - \sum_{i=1}^n z_i$. Then $z_1 \geq 0, \ldots, z_n \geq 0, z_{n+1} \geq 0$ and $\sum_{i=1}^{n+1} z_i = 1$. Thus, the transformation of variables allows us to substitute the problem (4.6) for the following problem of global minimization over the unit simplex $S_* \subset \mathbb{R}^{n+1}$:

$$\text{minimize} \quad g(z_1, \ldots, z_n) \quad \text{subject to } z = (z_1, \ldots, z_n, z_{n+1}) \in S_*.$$

This problem can be solved by the cutting angle method.

We now return to the problem (3.6). Consider a finite set $A = \{a^i : i = 1, \dots, r\}$. Let

$$\bar{a}_i = \min_{j \le r} a_i^j, \quad \bar{b}_i = \max_{j \le r} a_i^j, \quad i = 1, \dots, n.$$

Let x^* be a solution to the problem (3.6). We can assume that $\bar{a}_i \le x_i^* \le \bar{b}_i,\ i = 1, \dots, n$. Using the above technique we can transform the problem (3.6) to a problem of global minimization over the unit simplex in $\mathbb{R}^{n+1}$. Recall that the objective function f_t in this problem is Lipschitz continuous with Lipschitz constant $L = r$. It follows from Proposition 3.1 that there exists $\bar{c}^t > 0$ such that $f_t(x) \ge \bar{c}^t$ for all $x \in \mathbb{R}^n$. Thus $c' = 2r - \bar{c}^t$, where c' is a constant given by (4.4).

As mentioned above the most difficult part of the cutting angle method is minimization of the function h_j defined by (4.5) over the unit simplex. If the number j of min-type functions is fairly large, then the solution of this problem is time-consuming. The cutting angle method produces quickly several first iterations and much more time is required for the next iterations. This observation leads to a fruitful combination of the cutting angle method and a local search. Then the cutting angle method is exploited in order to leave a stationary point calculated by a local method.

For a local search we shall exploit methods of local nonsmooth optimization described in the previous section. These methods for each initial point $\bar{y}$ construct a sequence $\{y^k\}$ such that

1) each limit point of the sequence $\{y^k\}$ is a stationary point of f;

2) the sequence $\{f(y^k)\}$ is strictly decreasing: $f(y^{k+1}) < f(y^k)$ for all k.

Starting from an arbitrary point $\bar{y}^0$ we use a local search in order to obtain a stationary point y^1. If y^1 is not a global minimizer then the cutting angle method leads to a point $\bar{y}^1$ such that $f(\bar{y}^1) < f(y^1)$. Starting a local search from the point $\bar{y}^1$ we obtain a new stationary point y^2 such that $f(y^2) < f(\bar{y}^1) < f(y^1) < f(\bar{y}^0)$ and so on.

Applying the cutting angle method to the minimization of the function f we are not able to take into account the known value of this function at a stationary point y, which was found by a local search. In order to use this value we shall transform the function f to a new function ψ. The choice of the transformed function ψ is discussed in detail in [21]. In the present paper we shall consider the following function ψ as an objective function of a global optimization problem:

$$\psi(x) = \min_{k=1,\dots,n} \min_{\alpha \in A_k} \min(f(y), f(\alpha x + (1-\alpha)e^k)), \tag{4.7}$$

where $1 \in A_k \subset [0, 1]$, $k = 1, \dots, n$. Here y is a point, which has already been found by a local search.

Consider the following problem of global optimization:

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in S \tag{4.8}$$

where the function f is Lipschitz continuous. We propose the following algorithm for its solution which is based on the combination of the cutting angle method and a local search.

Algorithm 4.1 Algorithm for a global optimization

Step 0. *Initialization.* Choose an arbitrary starting point $\bar{y}^0$. Set $i = 0$.

Step 1. Find a stationary point of f over S by the local method, starting from the point $\bar{y}^i$. Denote this stationary point by y^i and let $f_i = f(y^i)$.

Step 2. Construct a transformed function ψ of the function f with respect to the point y^i and find a constant c^i such that $c^i > 2L_i - \min_{x \in S} \psi(x)$, where L_i is a Lipschitz constant of ψ.

Step 3. Take points $x^k = e^k$, $k = 1, \ldots, n, x^{n+1} = y^i$. Let

$$l^k = \frac{\psi(x^k) + c^i}{x^k},\ k = 1, \ldots, n+1$$

and set $j = n + 1$. Construct the function h_j, defined by

$$h_j(x) = \max_{k \le j} \min_{i \in I(l^k)} l_i^k x_i.$$

Step 4. Solve the problem

$$\text{minimize} \quad h_j(x) \quad \text{subject to} \quad x \in S. \tag{4.9}$$

Step 5. Let x^* be a solution to the problem (4.9). Set $j = j + 1$ and $x^j = x^*$.

Step 6. Compute $\psi^* = \psi(x^*)$. If $\psi^* < f_i$ then set $i = i + 1, \bar{y}^i = x^*$ and go to Step 1.

Step 7. Otherwise compute $l^j = \psi(x^j)/x^j$, define the function

$$h_j(x) = \max\{h_{j-1}(x), \min_{i \in I(l^j)} l_i^j x_i\} \equiv \max_{k \le j} \min_{i \in I(l^k)} l_i^k x_i$$

and go to Step 4.

Proposition 4.2 *(see [21]) Assume that the function f has a finite number of stationary points. Then Algorithm 4.1 terminates after a finite number of iterations at a global minimizer of f.*

Remark 4.1 The choice of the sets $A_k, k = 1, \ldots, n$ depends on the problem under consideration and in particular, on the number of variables. The number of elements of A_k should be large enough in order to obtain a good minorant for the objective function f. On the other hand it should not be too large, otherwise we will have a large number of objective function evaluations at each iteration of the cutting angle method. Numerical experiments show that the best choice in this situation is to consider sets A_k, which contain 4-7 points. In our numerical experiments A_k consists of 5 elements and $A_k = \{0.2, 0.4, 0.6, 0.8, 1\}$ for all $k = 1, \ldots, n$.

5 Conclusion

1. We have considered a nonsmooth optimization approach to clustering problems, which is based on the unconstrained minimization of the cluster function. We suggest using the cluster function as the measure of fitness of a collection of points to be the centres of clusters.

2. We have suggested two approaches to the choice of initial points for application of local optimization technique (Discrete Gradient methods). One of them significantly reduce the number of variables in clustering problems. This algorithm calculates clusters step by step, so it allows one to easily vary the number of clusters according to the criteria suggested by the nature of the decision making situation whilst not incurring the obvious costs of the increased complexity of the solution procedure. The suggested approach utilizes a stopping criterion that prevents the appearance of small and artificial clusters. Results of the numerical experiments presented in this paper show that the algorithm achieves better results than the k-means algorithm, comparable and sometimes even better results than many metaheuristics of global optimization using significantly less computational time.

   The other approach suggests the use of different versions of the k-means method for the search of an initial point.

3. We have introduced an approach for the definition of the structure of clusters and demonstrate that optimization starting from a point obtained by the k-means method can significantly improve the structure of clusters.

4 The usage of the cluster functions for the search of centre of clusters allows significant reduction of the number of instances in a dataset without destroying the structure of this dataset. This aspect is extremely important for clustering in large scale datasets. We suggest two approaches to the cleaning of large scale datasets. Numerical experiments confirm the efficiency of these approaches.

5. We have considered an approach to classification that is based on non-smooth and global optimization. Classes in the database under consideration are approximated by using cluster centers in these classes. So the cluster analysis problem is solved for each class. A feature selection algorithm and optimization algorithms for the step-by-step search of clusters were suggested. To verify the effectiveness of the proposed approach numerical experiments using several real-world databases have been carried out.

6. The proposed algorithms are effective for solving classification problems at least for databases considered in this paper. The version of the main Algorithm 3.2 with local optimization based on continuous approximation to the quasidifferential, produce results close to the best results obtained by different methods. The version of this algorithm with global optimization provided the best results for many datasets under consideration. Thus the global optimization is an effective tool for solving classification problems.

7. Methods based on the continuous approximation to the quasidifferential are better than those based on the continuous approximation to the Clarke subdifferential. At the same time the former requires more computational efforts and CPU time.

8. The feature selection stage is an important part of the algorithm. Results of numerical experiments show that in some instances the subset of attributes calculated by this algorithm provide better approximation of the test set than the full set of attributes.

9. The local versions of the main algorithm for classification (Algorithms 1 and 2) can be used when one wants to achieve high accuracy for a reasonable computational efforts and CPU time and finally. The global version of the main algorithm (Algorithm 3) can be used when one wants to obtain classification with very high accuracy neglecting computational efforts and CPU time.

10. For local optimization a derivative free discrete gradient method has been applied. A specific form of the cluster function allows the use of methods based on the continuous approximations to both Clarke subdifferential and Demyanov-Rubinov quasidifferential. Global optimization was carried out by a combination of the cutting angle method and a local search.

6 Appendix: Datasets, which were used for numerical experiments

Here we give a very short description of datasets that were used for numerical experiments. All these datasets, except *German towns*,*Postal zones* and *Emergency* can be found in [70]. Description of the *German towns* and *Postal zones* datasets see in [86]. The *Emergency* dataset was created in the emergency department on one of Australian hospitals.

Australian credit This dataset has been studied by J.R. Quinlan (see [73]). The purpose of this dataset is to devise rules for deciding on credit risk. Interpretation of the results is made difficult because the attributes and classes have been coded to preserve confidentiality. There are 2 classes and 690 observations in the dataset, 383 of them belong to the first class and 307 to the second one. In the Australian credit dataset features 1, 4, 6, 8, 9, 10, 11, 12 are qualitative and features 2, 3, 5, 7, 13, 14 are continuous.

Diabetes This dataset was originally donated by Vincent Sigillito, Applied Physics Laboratory, John Hopkins University, Laurel, USA and was constructed by constrained selection from a larger database held by the National Institute of Diabetes and Digestive and Kidney Diseases. It contains 2 classes, 768 observations and 8 attributes.

Emergency Originally this dataset contains 15193 observations, 7 features. All features in this dataset are numerous. The subset of features (1,2,3,5) was obtained by feature selection algorithm, and we used this subset in our research. We observed 7 points from the Emergency dataset, with a very large first coordinate or the coordinate 3 (which represents the age) was more than 200. We supposed, that these observations were questionable and removed them from the dataset. The new dataset contains 15186 observations.

German towns and **Postal zones**. The following three datasets are used as test datasets for clustering:

1. 'German towns', which uses the Cartesian coordinates of 59 towns and has 59 records with 2 attributes.

2. 'Bavaria 1' contains 89 postal zones in Bavaria (Germany) and their names. It contains 89 records with 3 attributes.

3. 'Bavaria 2' is the above Bavarian postal zones but with four attributes which are the number of self-employed people, of civil servants, clerks and manual workers. The number of patterns is again 89.

Heart disease This dataset contains data on the presence or absence of heart disease given the results of various medical tests carried out on a patient. This database comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano, M.D. PhD of V.A. Medical Centre, Long Beach, C.A. It is a part of the collection of databases at the University of California, Irvine, collated by David Aha. It contains 2 classes and has 297 observations. 160 of them are from the first class and 137 are from the second one. In the Heart disease dataset, features 2, 6, 9, 12 are qualitative and the features 1, 3, 4, 5, 7, 8, 10, 13 are continuous.

Letters The dataset was introduced by David Stale. It based on various fonts representation. The dataset consists of 20000 observations, 26 classes, 16 numerical attributes.

Liver-disorder This dataset was donated by Richard S. Forsyth BUPA Medical research Ltd. It contains 2 classes, 345 observations and 6 attributes.

Pen-based recognition of handwritten digits (Pendig) This dataset was introduced by E. Alpaydin and Fevzi Alimoglu. It contains 10 classes, 10992 observations, 16 attributes. All input attributes are integers $1 \ldots 100$.

Satellite image (SatIm, image segmentation) The original data for this dataset was generated from data purchased from NASA. It contains 6435 observations, 6 classes, 36 numerical attributes. All attributes are continuous.

Vehicle This data was originally gathered at the Turing Institute in 1986-87 by J. P. Siebert. It contains 846 observations, 4 classes, 18 attributes.

Wisconsin Breast Cancer This dataset was created by W.H. Wolberg, General Surgery Department, University of Wisconsin, Clinical Sciences Center, W.N. Street and O.L. Mangasarian, Computer Sciences Department, University of Wisconsin. It contains 2 classes, 569 observations and 30 attributes.

References

[1] K.S. Al-Sultan, A tabu search approach to the clustering problem, *Pattern Recognition*, 28(9) (1995), 1443-1451.

[2] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., New York, NY, 1989.

[3] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, NY, 1973.

[4] J. Abello, P.M. Pardalos and M.G.C. Resende (eds.), *Handbook of Massive Datasets*, Kluwer Academic Publishers, Dordrecht, 2001.

[5] K.S. Al-Sultan and M.M. Khan, Computational experience on four algorthms for the hard clustering problem, *Pattern Recognition Letters*, **17**, 1996, 295-308.

[6] M.Yu. Andramonov, A.M. Rubinov and B.M. Glover, Cutting angle method in global optimization, *Applied Mathematics Letters*, 12, 1999, 95-100.

[7] G. P. Babu and M. N. Murty. A near optimal initial seed value selection in the k-meanws algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, October 1993.

[8] R. A. Baeza-Yates. Introduction to data structures and algorithms related to information retrieval. In W.B. Frakes and R. Baeza Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 13–27. Prentice Hall, Upper Saddle River, NJ, 1992.

[9] A.M. Bagirov, A method of approximating a subdifferential, *Russian Journal of Computational Mathematics and Mathematical Physics,* 32, 1992, 561-566.

[10] A.M. Bagirov and A.A. Gasanov, A method of approximating a quasidifferential, *Russian Journal of Computational Mathematics and Mathematical Physics,* 35, 1995, 403-409.

[11] A.M. Bagirov, Continuous subdifferential approximation and its construction, *Indian Journal of Pure and Applied Mathematics*, 1, 1998, 17-29.

[12] A.M. Bagirov, Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis, *Investigacao Operacional*, 19, 1999, 75-93.

[13] A.M. Bagirov, Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices, In: A. Eberhard et al. (eds.) *Progress in Optimization: Contribution from Australasia*, Kluwer Academic Publishers, 1999, 147-175.

[14] A.M. Bagirov, Numerical methods for minimizing quasidifferentiable functions: a survey and comparison, In: V.F. Demyanov and A.M. Rubinov (eds.), *Quasidifferentiability and Related Topics*, Kluwer Academic Publishers, 2000, 33-71.

[15] A.M. Bagirov, A method for minimzation of quasidifferentiable functions, *Optimization Methods and Software*, 17(1) (2002), 31-60.

[16] A.M. Bagirov, A.M. Rubinov and J. Yearwood, Using global optimization to improve classification for medical diagnosis and prognosis, *Topics in Health Information Management* 22(2001) 65-74.

[17] A.M. Bagirov, A.M. Rubinov and J. Yearwood, A heuristic algorithm for feature selection based on optimization techniques. In: R. Sarker, H. Abbas and C.S. Newton (eds.), *Heuristic and Optimization for Knowledge Discovery*, Idea Publishing Group, 2002.

[18] A.M. Bagirov, A.M. Rubinov and J. Yearwood, A global optimization approach to classification, *Optimization and Engineering*, 3, 2002, 129-155.

[19] A.M. Bagirov and A.M. Rubinov, Global minimization of increasing positively homogeneous function over unit simplex, *Annals of Operations Research,* 98, 2000, 171-187.

[20] A.M. Bagirov and A.M. Rubinov, Modified versions of the cutting angle method, In: N. Hadjisavvas and P.M. Pardalos(eds), *Advances in Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht, 2001, 245-268.

[21] A.M. Bagirov and A.M. Rubinov, The cutting angle method and a local search, *Journal of Global Optimization*, to appear.

[22] A.M. Bagirov and J. Yearwood, A new nonsmooth optimization algorithm for clustering problems, Research Report 03/02, University of Ballarat, Australia, 2003. Submitted to: *European Journal of Operational Research*.

[23] L. Batten and G. Beliakov, Fast algorithm for the Cutting Angle Method of Global Optimization, *Journal of Global Optimization*, vol.24 (2002), pp. 149-161.

[24] K.P. Bennett and O.L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, *Optimization Methods and Software*, 1, 1992, 23-34.

[25] H.H. Bock, *Automatische Klassifikation*, Vandenhoeck & Ruprecht, Gottingen, 1974.

[26] H.H. Bock, Clustering and neural networks, In: A.Rizzi, M. Vichi and H.H. Bock (eds), *Advances in Data Science and Classification*, Springer-Verlag, Berlin, 1998, 265-277.

[27] P.S. Bradley and O.L. Mangasarian, Feature selection via concave minimization and support vector machines, *"Machine Learning Proceedings of the Fifteenth International Conference (ICML'98)*, J. Shavlik (ed.), Morgan Kaufmann, San Francisco, California, 1998, 82-90.

[28] P.S. Bradley and O.L. Mangasarian, Massive data discrimination via linear support vector machines, *Optimization Methods and Software,* 13, 2000, 1-10.

[29] P.S. Bradley, Usama M. Fayyad and O.L. Mangasarian, Data mining: overview and optimization opportunities, *INFORMS Journal on Computing*, 11, 1999, 217-238.

[30] D.E. Brown and C.L. Entail, A practical application of simulated annealing to the clustering problem, *Pattern Recognition*, 25(1992), 401-412.

[31] M.Brown, W. Grundy, D. Lin, N. Christianini, C. Sugnet, T. Furey, M. Ares, and D. Haussler. Knowledg-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, 97:262–267, 2000.

[32] N. J. Bhuyan, V. V. Raghavan, and K. E. Venkatesh. Genetic algorithms for clustering with an ordered representation. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 408–415, 1991.

[33] G. Carpenter and S. Grossberg. Art3: Hierarchical search using chemical transmitters in self organising pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

[34] C. Chen and O.L. Mangasarian, Hybrid misclassification minimization, Mathematical Programming Technical Report, 95-05, University of Wisconsin, 1995.

[35] F.H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley, New York, 1983.

[36] D.DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1):161–190, 2002.

[37] V.F. Demyanov and A.M. Rubinov, *Constructive Nonsmooth Analysis*, Peter Lang, Frankfurt am Main, 1995.

[38] I.S. Dhillon, J. Fan and Y. Guan, Efficient clustering of very large document collections, In: *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publishers, Dordrecht, 2001.

[39] G. Diehr, Evaluation of a branch and bound algorithm for clustering, *SIAM J. Scientific and Statistical Computing*, 6(1985), 268-284.

[40] R. Dubes and A.K. Jain, Clustering techniques: the user's dilemma, *Pattern Recognition*, 8(1976), 247-260.

[41] Gavin Finnie and Zhaohao Sun. r^5 model for case-based reasoning. *Knowledge-Based Systems*, 16:59–65, 2003.

[42] D.B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks*, Vol. 5, Issue 1, 1994, pp. 3-14.

[43] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* 2nd edition. Academic Press, 1990.

[44] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., Inc., Redwood City,CA, 1989.

[45] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems Man and Cybernetics*, 1:122–128, Jan/Feb 1986.

[46] P. Hanjoul and D. Peeters, A comparison of two dual-based procedures for solving the p-median problem, *European Journal of Operational Research,* 20(1985), 387-396.

[47] P. Hansen and B. Jaumard, Cluster analysis and mathematical programming, *Mathematical Programming,* 79(1-3) (1997), 191-215.

[48] J.-P. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms,* Vol. 1 and 2, Springer-Verlag, Berlin, New York, 1993.

[49] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[50] D.M. Houkins, M.W. Muller and J.A. ten Krooden, Cluster analysis, In: *Topics in Applied Multivariate Analysis*, Cambridge University press, Cambridge, 1982.

[51] A.K. Jain, M.N. Murty and P.J. Flynn, Data clustering: a review, *ACM Computing Surveys* 31(3)(1999), 264-323.

[52] N. Jardine and R. Sibson. *Mathematical Taxonomy.* Wiley, London and New York, 1971.

[53] R.E. Jensen, A dynamic programming algorithm for cluster analysis, *Operations Research,* 17(1969), 1034-1057.

[54] T. Joachims. Text categorization with support vector machines: Learning with many relevant teatures. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer-Verlag.

[55] D. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 442–449, 1991.

[56] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[57] K.C. Kiwiel, *Methods of descent for nondifferentiable optimization, Lecture Notes in Mathematics,* 1133, Springer-Verlag, Berlin, 1985.

[58] T. Kohonen. *Self Organization and Associative Memory.* Springer Information Sciences Series. Springer-Verlag, 3 edition, 1989.

[59] W.L.G. Koontz, P.M. Narendra and K. Fukunaga, A branch and bound clustering algorithm, *IEEE Transactions on Computers*, 24(1975), 908-915.

[60] S. Y. Lu and K. S. Fu. A sentence to sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems Mans and Cybernetics*, 8:381–389, 1978.

[61] J. B. MacQueen, *Some Methods for Classification and Analysis of Multivariate observations.* In: L.M. LeCam and J. Neyman (eds.), Proceedings of the Firth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1. Berkeley: University of California Press, 1967.

[62] O.L. Mangasarian, Misclassification minimization, *Journal of Global Optimization*, 5, 1994, 309-323.

[63] O.L. Mangasarian, Mathematical programming in data mining, *Data Mining and Knowledge Discovery,* 1(1997)183-201.

[64] G.J. McLachlan, *Discriminat Analysis and Statistical Pattern Recognition.* New York, John Wiley, 1992.

[65] G.J. McLachlan, D. Peel, and P. Prado. Clustering via normal mixture models.

[66] J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1971.

[67] D. Michie, D.J. Spiegelhalter and C.C. Taylor (eds.), *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Series in Artificial Intelligence, London, 1994.

[68] R. Mifflin, Semismooth and semiconvex functions in constrained optimization, *SIAM Journal on Control and Optimization*, 15(6)(1977), 959-972.

[69] B. Mirkin, *Mathematical Classification and Clustering*, Kluwer Academic Publishers, 1996.

[70] P.M. Murphy and D.W. Aha, UCI repository of machine learning databases. Technical report, Department of Information and Computer science, University of California, Irvine, 1992. www.ics.uci.edu/mlearn/MLRepository.html.

[71] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms which use cluster centres. *Computer Journal*, 26:354–359, 1984.

[72] G. Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56:836–862, 1968.

[73] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, 1993.

[74] V. V. Raghavan and K. Birchand. A comparison of the stability characteristics of some graph theoretic clustering methods. In *Proceedings of the Second international Conference on Information Storage and Retreival*, pages 10–22, 1979.

[75] C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, London, 1993.

[76] A.M. Rubinov, *Abstract Convexity and Global Optimization,* Kluwer Academic Publishers, Dordrecht, 2000.

[77] A.M. Rubinov and N.V. Soukhoroukova, A nonsmooth optimization approach to clustering large-scale datasets, in preparation.

[78] A.M. Rubinov, N.V. Soukhoroukova and J. Yearwood, Clustering for studuing structure and quality of datasets, Research Report 01/24, University of Ballarat, 2001.

[79] A.M. Rubinov and J. Ugon, Skeletons of finite sets of points, submitted paper.

[80] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, Mass., 2002.

[81] S.Z. Selim and M.A. Ismail, k-means-type algorithm: generalized convergence theorem and characterization of local optimality, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1984), 81-87.

[82] S.Z. Selim and K.S. Al-Sultan, A simulated annealing algorithm for the clustering, *Pattern Recognition*, 24(10) (1991), 1003-1008.

[83] I. Sethi and A.K. Jain, editors. *Artificial Neural Networks and Pattern Recognition: Old and new Connections.* Elsevier Science, New York, NY, 1991.

[84] Yi Shang and Benjamin W. Wah. Global optimization for neural network training. *IEEE Computer*, 29(3):31–44, March 1996.

[85] P. H. A. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, London, UK, 1973.

[86] H. Spath, *Cluster Analysis Algorithms*, Ellis Horwood Limited, Chichester, 1980.

[87] L.X. Sun, Y.L. Xie, X.H. Song, J.H. Wang and R.Q. Yu, Cluster analysis by simulated annealing, *Computers and Chemistry*, 18(1994), 103-108.

[88] J. H. Jr. Ward. Hierarchical grouping to optimize and objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.