A Binary Linear Programming-Based K-Means Algorithm For Clustering with Must-Link and Cannot-Link Constraints

Philipp Baumann
Department of Business Administration, University of Bern, Schuetzenmattstrasse 14, 3012 Bern, Switzerland
e-mail: philipp.baumann@pqm.unibe.ch

Abstract—**Clustering is probably the most extensively studied problem in unsupervised learning. Traditional clustering algorithms assign objects to clusters exclusively based on features of the objects. Constrained clustering is a generalization of traditional clustering where additional information about a dataset is given in the form of constraints. It has been shown that the clustering accuracy can be improved substantially by accounting for these constraints. We consider the constrained clustering problem where additional information is given in the form of must-link and cannot-link constraints for some pairs of objects. Various algorithms have been developed for this specific clustering problem. We propose a binary linear programming-based k-means approach that can consider must-link and cannot-link constraints. In a computational experiment, we compare the proposed algorithm to the DILS$_{\text{CC}}$ algorithm, which represents the state-of-the-art. Our results on 75 problem instances indicate that the proposed algorithm delivers better clusterings than the DILS$_{\text{CC}}$ algorithm in much shorter running time.**

Keywords—**Constrained clustering, must-link, cannot-link, k-means, binary linear programming**

I. INTRODUCTION

Clustering is a fundamental task in machine learning. The goal is to assign objects of a dataset to clusters such that the members of each cluster are similar to each other in some way. In most practical clustering applications, some additional information about the dataset can be expressed in the form of constraints. Examples of such constraints are lower and upper bounds on the size of clusters, or must-link and cannot-link constraints for pairs of objects. It has been shown that the clustering accuracy can be improved substantially by considering these constraints during the clustering process. Examples of successful applications of constrained clustering can be found in various fields including robotics, computer vision, marketing, and homeland security.

We consider here the following constrained clustering problem. Given is a set of objects. Each object is described by numeric features and belongs to one of k clusters. The true cluster assignments are only used for evaluating the clustering accuracy and are not available as input parameters. Furthermore, must-link and cannot-link constraints are given for some pairs of objects. Two objects must be assigned to the same cluster if they are subject to a must-link constraint and two objects cannot be assigned to the same cluster if they are subject to a cannot-link constraint. The goal is to assign the objects to k clusters such that all must-link and cannot-link constraints are satisfied and the agreement of the cluster assignments with the true assignments is as large as possible. The agreement of the two assignments is measured by the Adjusted Rand Index (ARI) introduced by Hubert and Arabie [1], which ignores permutations and adjusts for chance.

Various algorithms have been proposed for constrained clustering with must-link and cannot-link constraints. Most of these algorithms treat the must-link and cannot-link constraints as soft constraints to reduce the complexity of finding a feasible partition. González-Almagro et al. [2] have recently performed a comprehensive computational comparison of seven algorithms on a diverse collection of data sets. Some of the tested algorithms treat must-link and cannot-link constraints as hard constraints and some treat them as soft constraints. One of the tested algorithms, the dual iterative local search algorithm (DILS$_{\text{CC}}$), was newly introduced in the same paper. The other six algorithms were established state-of-the-art algorithms. A statistical analysis of the results provided strong evidence that the new DILS$_{\text{CC}}$ algorithm performs best in terms of ARI values and therefore represents the new state-of-the-art algorithm. DILS$_{\text{CC}}$ is a population-based metaheuristic that keeps only two individuals in memory at all times. Recombination and mutation operations are applied to the two individuals to escape from local optima. An iterative local search procedure is used to improve the two individuals after modifications. A drawback of the DILS$_{\text{CC}}$ algorithm is that it requires relatively long running times to deliver satisfactory ARI values.

In this paper, we introduce the BLPKM$_{\text{CC}}$ algorithm for clustering with must-link and cannot-link constraints. The BLPKM$_{\text{CC}}$ algorithm alternates between an object-assignment step and a cluster-center-update step like the standard k-means algorithm. The assignment step of the algorithm is formulated as a binary linear program and solved with a mathematical programming solver. The binary linear program of the assignment step ensures that must-link and cannot-link constraints are respected when assigning objects to clusters. We use the same 75 instances as González-Almagro et al. [2] to compare the proposed BLPKM$_{\text{CC}}$ algorithm to the DILS$_{\text{CC}}$ algorithm. It turns out that the BLPKM$_{\text{CC}}$ algorithm outperforms the DILS$_{\text{CC}}$ algorithm by a surprisingly large margin, both in terms of ARI values and in terms of running time.

The paper is structured as follows. In Section II, we formally describe the constrained clustering problem. In Section III, we give a brief overview of state-of-the-art algorithms. In Section IV, we introduce the BLPKM$_{\text{CC}}$ algorithm. In Section V, we conduct the computational comparison between the BLPKM$_{\text{CC}}$ and the DILS$_{\text{CC}}$ algorithm. Finally, in Section VI, we conclude and provide some directions for future research.

II. CONSTRAINED CLUSTERING PROBLEM

We consider the following constrained clustering problem. Given is a dataset that contains n objects $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$. Each object $\boldsymbol{x}_i$ is a vector of d numeric features. The goal is to assign each object a label $l_i \in \{1, \ldots, k\}$ such that the dataset is partitioned into a predefined number of k clusters. Furthermore, a set of must-link (ML) and a set of cannot-link (CL) constraints are given. Both types of constraints are defined for pairs of objects (i, i'). A must-link constraint states that objects i and i' have the same label. A cannot-link constraint states that objects i and i' have different labels. The must-link and cannot-link constraints can be treated as soft constraints, i.e., a feasible assignment does not have to satisfy all of these constraints. The quality of a partition is measured by the Adjusted Rand Index (ARI). The ARI requires a true label $y_i \in \{1, \ldots, k\}$ for each object. The ARI takes as inputs the true labels (y_i) and the labels assigned by the clustering algorithm (l_i) and outputs a value in the interval [-1, 1]. A value close to 1 means a high agreement between the respective partitions and a value close to 0 means a low agreement. A value below zero means that the agreement is smaller than the expected agreement that would result when comparing the true partition to a random partition.

III. RELATED LITERATURE

Most existing algorithms incorporate the must-link and cannot-link constraints as soft constraints, i.e., violations of must-link and cannot-link constraints are tolerated, but penalized in the objective or fitness function. Examples of algorithms with soft must-link and cannot-link constraints are the TVClust and the RDPM algorithm of Khashabi et al. [3], the LCVQE algorithm of Pelleg and Baras [4], the CECM algorithm of Antoine et al. [5], the BRKGA+LS algorithm of de Oliveira et al. [6], and the DILS$_{\text{CC}}$ algorithm of González-Almagro et al. [2]. Relatively few algorithms treat the must-link and cannot-link constraints as hard constraints because it is known that the feasibility problem for clustering with must-link and cannot-link constraints is NP-complete (see Davidson and Ravi [7]). Examples of algorithms with hard must-link and cannot-link constraints are the COP-kmeans algorithm of Wagstaff et al. [8] and the binary optimization algorithm of Le et al. [9]. The recent computational comparison of González-Almagro et al. [2], which included algorithms with hard and algorithms with soft constraints, showed that the DILS$_{\text{CC}}$ algorithm can be considered the state-of-the-art algorithm in terms of ARI values. However, even for small datasets the running time of the DILS$_{\text{CC}}$ algorithm is quite high.

Recently, heuristics that use mathematical programming solvers have shown to deliver leading performance in short running times for capacitated clustering problems (cf., e.g., Stefanello et al. [10], Baumann [11], Gnägi and Baumann [12]). To investigate if such heuristics are competitive for clustering with must-link and cannot-link constraints, we propose a heuristic that uses the same problem decomposition scheme as the heuristic of Baumann [11] but considers must-link and cannot-link constraints.

IV. BLPKM$_{\text{CC}}$ ALGORITHM

The proposed BLPKM$_{\text{CC}}$ algorithm alternates between between an object-assignment step and a cluster-center-update step until a stopping criterion is reached. The assignment step is performed by solving a binary linear program which allows to easily integrate side constraints. Here, we formulate the binary linear program such that must-link and cannot-link constraints are accounted for. The main steps of the proposed algorithm are:

(1) The positions of the k cluster centers $\bar{\boldsymbol{c}}_j;\ j = 1, \ldots, k$ are determined by randomly selecting k distinct objects in the dataset. The initial cluster centers are placed at the positions of the selected objects.

(2) The objects are assigned to the clusters by solving the binary linear program BLP given below; Table I contains the relevant notation.

$$\text{BLP}\begin{cases}\text{Min.} \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{k} d_{ij} y_{ij} & & (1)\\ \text{s.t.} \displaystyle\sum_{j=1}^{k} y_{ij} = 1 & (i = 1, \ldots, n) & (2)\\ \displaystyle\sum_{i=1}^{n} y_{ij} \geq 1 & (j = 1, \ldots k) & (3)\\ y_{ij} = y_{i'j} & ((i, i') \in ML;\ j = 1, \ldots, k) & (4)\\ y_{ij} + y_{i'j} \leq 1 & ((i, i') \in CL;\ j = 1, \ldots, k) & (5)\\ y_{ij} \in \{0, 1\} & (i = 1, \ldots, n;\ j = 1, \ldots, k) & (6)\end{cases}$$

The objective function (1) minimizes the total distance between each object and the center of the cluster to which it is assigned to. Constraints (2) ensure that each object is assigned to exactly one cluster, and constraints (3) guarantee that each cluster contains at least one object. Constraints (4) and (5) ensure the must-link and cannot-link constraints, respectively.

(3) The centers of the clusters are updated. Let $C_j = \{i \in \{1, \ldots, n\} | y_{ij} = 1\}$ be the objects with label j in the solution of the BLP. The updated cluster center is then computed as $\bar{\boldsymbol{c}}_j = \frac{\sum_{i \in C_j} \boldsymbol{x}_i}{|C_j|}$.

Steps (2) and (3) are repeated as long as the total distance between each object and the center of the cluster to which it is assigned to (computed after step (3)) can be decreased. Figure 1 visualizes the application of the proposed algorithm to a toy problem instance. The subplots show the result of step (2) in each of the three iterations. The bottom right subplot shows that the final assignment coincides with the true assignment resulting in an optimal ARI value of 1.

V. COMPUTATIONAL COMPARISON

A. Datasets

We use the same 25 datasets as González-Almagro et al. [2]. Table II provides the number of objects, the number of features, the number of classes, the type (real-world or artificially

TABLE I: Notation

Sets	
ML	Pairs of objects subject to a must-link constraint
CL	Pairs of objects subject to a cannot-link constraint
Parameters	
n	Number of objects
k	Number of clusters
d_{ij}	Euclidean distance between object $\boldsymbol{x}_i$ and center $\bar{\boldsymbol{c}}_j$
Binary decision variables	
y_{ij}	=1, if object i is assigned to cluster j; =0, otherwise

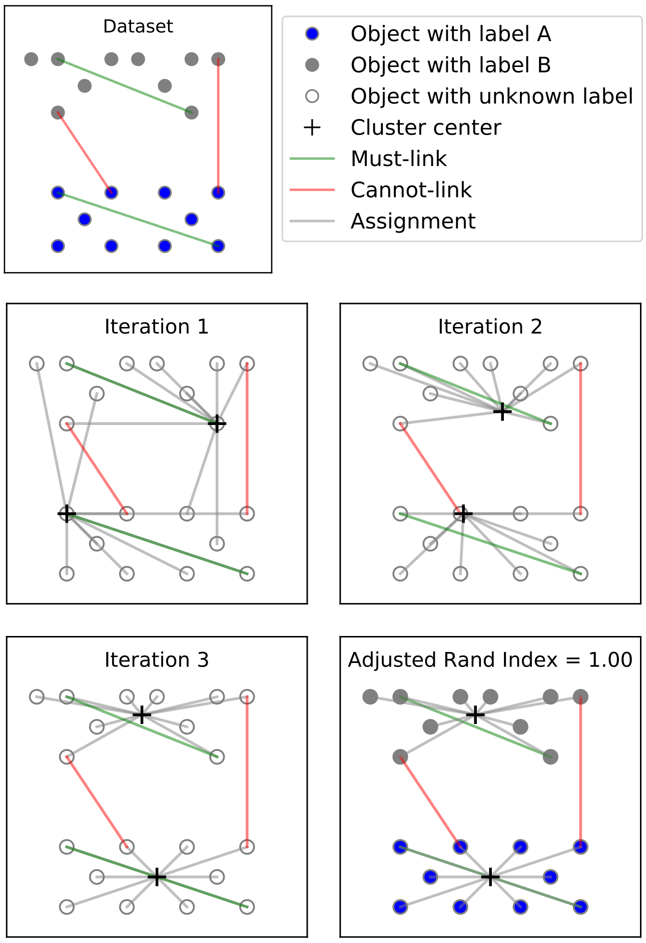

Fig. 1: Applying the BLPKM$_{\text{CC}}$ algorithm to a toy problem instance

generated), and the source of each dataset (Keel repository[1], Sklearn repository[2], GitHub repository[3], UCI repository[4]). We did not perform any preprocessing of these datasets with one exception. In the dataset Saheart, we replaced the two categories `Present` and `Absent` of the binary feature "Famhist" with one and zero, respectively.

[1]https://sci2s.ugr.es/keel/category.php?cat=clas
[2]https://scikit-learn.org/stable/datasets/index.html
[3]https://github.com/GermangUgr/DILS_CC
[4]https://archive.ics.uci.edu/ml/datasets.php

TABLE II: Characteristics of data sets

	Objects	Features	Classes	Type	Source
Appendicitis	106	7	2	real	Keel
Breast Cancer	569	30	2	real	Sklearn
Bupa	345	6	2	real	Keel
Circles	300	2	2	generated	GitHub
Ecoli	336	7	8	real	Keel
Glass	214	9	6	real	Keel
Haberman	306	3	2	real	Keel
Hayesroth	160	4	3	real	Keel
Heart	270	13	2	real	Keel
Ionosphere	351	33	2	real	Keel
Iris	150	4	3	real	Keel
Led7Digit	500	7	10	real	Keel
Monk2	432	6	2	real	Keel
Moons	300	2	2	generated	GitHub
Movement Libras	360	90	15	real	Keel
Newthyroid	215	5	3	real	Keel
Saheart	462	9	2	real	Keel
Sonar	208	60	2	real	Keel
Spectfheart	267	44	2	real	Keel
Spiral	300	2	2	generated	GitHub
Soybean	47	35	4	real	UCI
Tae	151	5	3	real	Keel
Vehicle	846	18	4	real	Keel
Wine	178	13	3	real	Keel
Zoo	101	16	7	real	Keel

B. Constraint sets

All datasets are classification datasets which means that the true class label is known for each object. González-Almagro et al. [2] used the true class labels to generate three sets of constraints for each of the 25 datasets with the method proposed in Wagstaff et al. [8]. The three sets differ with respect to the number of constraints and are referred to as CS$_{10}$, CS$_{15}$, and CS$_{20}$. Since the constraint sets are generated based on the true class labels, it is guaranteed that there is always a feasible solution. All constraint sets are publicly available here[5].

C. Experimental design

Since there are three constraint sets for each dataset, the test set comprises 75 problem instances in total. For the sake of comparability, we chose the same experimental design as González-Almagro et al. [2]. We applied the BLPKM$_{\text{CC}}$ algorithm 30 times to all 75 instances and report the average ARI value and the average running time per instance. With 75 instances and 30 runs per instance, we performed 2250 clusterings with our algorithm. We implemented the BLPKM$_{\text{CC}}$ in Python 3.7 and used Gurobi 8.1.1 as solver. Our implementation is available on GitHub[6]. All computations were executed on an Intel Core i7 2.2 GHz processor with 16 GB of RAM. The results of the DILS$_{\text{CC}}$ algorithm are those reported in González-Almagro et al. [2]. González-Almagro et al. [2] ran their algorithm on a computer cluster with 51 computing nodes.

[5]https://drive.google.com/drive/u/1/folders/1sjnPYitey8q9zrPKa_YpFS7iNKTT15QH
[6]https://github.com/phil85/BLPKM-CC

D. Numerical results

We first analyze the quality of the clusterings obtained by the two algorithms. Table III shows for each instance (combination of dataset and constraint set) the average ARI value obtained by the BLPKM$_{\text{CC}}$ and the DILS$_{\text{CC}}$ algorithm. For each instance, we highlighted the highest average ARI value in bold. The last row of the table contains the average values of the respective columns. We can conclude that the BLPKM$_{\text{CC}}$ algorithm outperforms the DILS$_{\text{CC}}$ algorithm by a surprisingly large margin. With the largest constraint set, the BLPKM$_{\text{CC}}$ algorithm obtains the maximum possible ARI value for 14 out of 25 instances. The quality of the clusterings of both algorithms increases with increasing size of the constraint set.

Next, we analyze the running time of the two algorithms. Table IV provides for each instance the average running time required by the BLPKM$_{\text{CC}}$ algorithm and the DILS$_{\text{CC}}$ algorithm to perform one run. For each instance, we highlighted the shortest average running time. The last row of the table contains the sums of the respective columns. The reported running times demonstrate that the BLPKM$_{\text{CC}}$ algorithm is substantially faster than the DILS$_{\text{CC}}$ algorithm across data- and constraint sets.

VI. CONCLUSIONS

We proposed a version of the BLPKM algorithm of Baumann [11] that can consider must-link and cannot-link constraints. In a comprehensive computational experiment based on 75 problem instances, the proposed algorithm outperforms the state-of-the-art algorithm both in terms of solution quality and running time. The implementation of the proposed algorithm is publicly available. In future research, we will develop initialization methods that identify promising starting positions for the cluster centers by taking into account the must-link and cannot-link constraints. Furthermore, we plan to perform a full factorial experimental analysis to study the impact of varying characteristics of problem instances such as the number of objects, the number of features, the number of classes on the performance of the proposed algorithm. Finally, we would like to evaluate the performance of the proposed algorithm when using the open-source solver SCIP instead of the commercial Gurobi solver.

REFERENCES

[1] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[2] G. González-Almagro, J. Luengo, J.-R. Cano, and S. García, "DILS: constrained clustering through dual iterative local search," *Computers & Operations Research*, p. 104979, 2020.

[3] D. Khashabi, J. Wieting, J. Y. Liu, and F. Liang, "Clustering with side information: From a probabilistic model to a deterministic algorithm," *arXiv preprint arXiv:1508.06235*, 2015.

[4] D. Pelleg and D. Baras, "K-means with large and noisy constraint sets," in *European Conference on Machine Learning*, 2007, pp. 674–682.

[5] V. Antoine, B. Quost, M.-H. Masson, and T. Denoeux, "CECM: Constrained evidential c-means algorithm," *Computational Statistics & Data Analysis*, vol. 56, pp. 894–914, 2012.

[6] R. M. de Oliveira, A. A. Chaves, and L. A. N. Lorena, "A comparison of two hybrid methods for constrained clustering problems," *Applied Soft Computing*, vol. 54, pp. 256–266, 2017.

[7] I. Davidson and S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 2005, pp. 138–149.

[8] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 577–584.

[9] H. M. Le, A. Eriksson, T.-T. Do, and M. Milford, "A binary optimization approach for constrained k-means clustering," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 383–398.

[10] F. Stefanello, O. C. de Araújo, and F. M. Müller, "Matheuristics for the capacitated p-median problem," *International Transactions in Operational Research*, vol. 22, pp. 149–167, 2015.

[11] P. Baumann, "A binary linear programming-based k-means approach for the capacitated centered clustering problem," in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2019, pp. 335–339.

[12] M. Gnägi and P. Baumann, "A matheuristic for large-scale capacitated clustering," *Under review*, 2020.

TABLE III: Average ARI values

	CS$_{10}$		CS$_{15}$		CS$_{20}$	
Dataset	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$
Appendicitis	0.573	**0.611**	**1.000**	0.957	**1.000**	**1.000**
Breast Cancer	**0.979**	0.755	**1.000**	0.792	**1.000**	0.796
Bupa	**0.931**	0.889	**1.000**	0.993	**1.000**	0.988
Circles	**0.850**	0.781	**1.000**	**1.000**	**1.000**	**1.000**
Ecoli	**0.686**	0.039	**0.912**	0.091	**0.974**	0.264
Glass	**0.286**	0.008	**0.763**	0.076	**0.942**	0.258
Haberman	**0.929**	0.802	**1.000**	**1.000**	**1.000**	**1.000**
Hayesroth	**0.173**	0.057	**0.978**	0.478	**0.923**	0.816
Heart	**0.885**	0.846	**1.000**	**1.000**	**1.000**	**1.000**
Ionosphere	**0.943**	0.809	**1.000**	0.973	**1.000**	0.984
Iris	**0.584**	0.550	0.598	**0.832**	0.574	**0.953**
Led7Digit	**0.611**	0.013	**0.877**	0.012	**0.988**	0.017
Monk2	**0.963**	0.823	**1.000**	0.899	**1.000**	0.899
Moons	**0.987**	0.963	**1.000**	**1.000**	**1.000**	**1.000**
Movement Libras	**0.312**	0.019	**0.348**	0.018	**0.503**	0.020
Newthyroid	**0.865**	0.040	**0.984**	0.390	**0.984**	0.845
Saheart	**0.983**	0.788	**1.000**	0.870	**1.000**	0.867
Sonar	**0.743**	0.710	0.981	**0.981**	**1.000**	**1.000**
Soybean	**0.607**	0.289	**0.607**	0.468	**0.805**	0.629
Spectfheart	0.871	**0.895**	**1.000**	**1.000**	**1.000**	**1.000**
Spiral	**0.857**	0.849	**1.000**	**1.000**	**1.000**	**1.000**
Tae	**0.046**	0.028	**0.547**	0.386	**0.982**	0.846
Vehicle	**0.956**	0.023	**1.000**	0.066	**1.000**	0.171
Wine	**0.397**	0.326	0.536	**0.740**	0.583	**0.898**
Zoo	**0.629**	0.221	**0.788**	0.193	**0.819**	0.250
Mean	**0.706**	0.485	**0.877**	0.649	**0.923**	0.740

TABLE IV: Average running time to perform one run in seconds

	CS$_{10}$		CS$_{15}$		CS$_{20}$	
Dataset	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$	BLPKM$_{\text{CC}}$	DILS$_{\text{CC}}$
Appendicitis	**0.2**	836.5	**0.1**	870.4	**0.1**	941.6
Breast Cancer	**0.5**	8,572.7	**0.5**	9,223.6	**0.8**	10,625.8
Bupa	**0.2**	3,222.8	**0.2**	3,482.5	**0.3**	3,934.5
Circles	**0.3**	2,491.1	**0.2**	2,673.5	**0.2**	3,016.7
Ecoli	**2.4**	2,655.8	**2.5**	2,890.1	**2.3**	3,376.0
Glass	**0.6**	1,657.8	**0.8**	1,766.6	**0.8**	1,956.7
Haberman	**0.2**	2,639.8	**0.1**	2,867.3	**0.2**	3,274.8
Hayesroth	**0.4**	1,211.5	**0.2**	1,266.8	**0.2**	1,390.4
Heart	**0.2**	2,586.0	**0.1**	2,776.3	**0.2**	3,080.3
Ionosphere	**0.2**	4,392.8	**0.2**	4,688.9	**0.2**	5,272.5
Iris	**0.2**	1,128.8	**0.2**	1,180.9	**0.2**	1,303.0
Led7Digit	**4.2**	4,101.8	**5.0**	4,622.6	**4.7**	5,550.2
Monk2	**0.3**	4,255.2	**0.2**	4,643.2	**0.4**	5,394.4
Moons	**0.2**	2,483.2	**0.1**	2,679.2	**0.2**	3,004.0
Movement Libras	**5.4**	3,403.7	**7.3**	3,724.6	**14.5**	4,146.8
Newthyroid	**0.3**	1,709.6	**0.2**	1,820.4	**0.3**	2,074.7
Saheart	**0.2**	4,929.8	**0.3**	5,437.3	**0.4**	6,176.6
Sonar	**0.2**	2,513.6	**0.1**	2,623.1	**0.1**	2,848.1
Soybean	**0.2**	367.3	**0.2**	375.3	**0.2**	402.4
Spectfheart	**0.2**	3,449.8	**0.2**	3,660.1	**0.2**	3,978.0
Spiral	**0.2**	2,490.6	**0.2**	2,681.7	**0.2**	3,027.2
Tae	**0.3**	1,158.9	**0.3**	1,206.9	**0.2**	1,321.6
Vehicle	**1.3**	10,269.1	**1.4**	11,791.3	**2.5**	14,368.6
Wine	**0.2**	1,486.7	**0.2**	1,564.4	**0.2**	1,724.4
Zoo	**0.3**	744.8	**0.3**	769.8	**0.4**	843.4
Sum	**18.8**	74,759.8	**21.2**	81,286.9	**29.9**	93,032.6