Control and Cybernetics

vol. **45** (2016) No. 4

The PM-M prototype selection system*

by

Karol Grudziński

Department of Physics, Kazimierz Wielki University,
Aleja Powstańcw Wielkopolskich 2, 85-090 Bydgoszcz, Poland
grudzinski.k@gmail.com

Abstract: In this paper, the algorithm, realizing the author's prototype selection method, called PM-M (Partial Memory - Minimization) is described in details. Computational experiments that have been carried out with the raw PM-M model and with its majority ensembles indicate that even for the system, for which the average size of the selected prototype sets constitutes only about five percent of the size of the original training datasets, the obtained results of classification are still in a good statistical agreement with the 1-Nearest Neighbor (IB1) model which has been trained on the original (i.e. unpruned) data. It has also been shown that the system under study is competitive in terms of generalization ability with respect to other well established prototype selection systems, such as, for example, CHC, SSMA and GGA. Moreover, the proposed algorithm has shown approximately one to three orders of magnitude decrement of time requirements with respect to the necessary time, needed to complete the calculations, relative to the reference prototype classifiers, taken for comparison. It has also been demonstrated that the PM-M system can be directly applied to analysis of very large data unlike most other prototype methods, which have to rely on the stratification approach.

Keywords: selection of reference instances, prototype selection, k-Nearest Neighbors algorithm, classification of data

1. Introduction

Selection of prototype cases (instances, vectors) is a very important subfield of machine learning. Selection of prototype instances constitutes a family within a broader approach, known as reference vectors selection, training data compression or training data pruning (Garcia et al, 2011; Grochowski and Jankowski, 2004; Jankowski and Grochowski, 2004). Instances can be selected from a training set for a variety of reasons. The fundamental case is provided by the need of finding a very low number of interesting samples in data, such that carry a particularly large amount of information. Another reason for data pruning may be the necessity to filter noisy samples or outliers from a training set. One may also need

*Submitted: February 2012; Accepted: February 2017.

to reduce the size of a training set to speed up learning and instead of drawing a random stratified sample from training data a reference vectors selection approach can be used. If a classifier employed is a minimal-distance one, pruning of data may also significantly reduce the time requirements for the testing phase (i.e. classification of unseen samples). Prototype selection is a necessary step in extraction of prototype-based rules (Duch and Blachnik, 2004; Duch and Grudzinski, 2001). The importance of reference vectors selection methods in application to data analysis cannot be overestimated. In Grudzinski (2004) we described our PM-M prototype system and gave some preliminary numerical results. In this paper we provide for the first time the algorithm for the PM-M system. We also study its generalization ability on a much larger number of datasets and give new numerical results that justify the usefulness of this method.

The PM-M algorithm evolved in time from a simple implementation, which initially had only been a part of our Similarity Based Learner software (Grudzinski and Duch, 1996-2017), to the model being also a part of WEKA workbench (Hall et al., 2009), modified by us and called SuperWeka (Grudzinski, 2006-2017).

In order to establish a general idea behind the PM-M system, imagine that to each of the training cases a binary weight has to be assigned, indicating whether a sample should be retained or not. Thus, if n is the number of training samples, we have the n-dimensional vector of parameters to optimize. A downhill simplex method (Nelder and Mead, 1965) for function minimization is then used to find this optimal set of parameters. The cost function returns the root mean squared error that the underlying arbitrary classifier makes either by performing cross-validation on a pruned training partition (the first variant of the method) or by training it on a compressed training partition and evaluating it on the whole original training set (the second variant).

Despite employing the simplex minimization method which is often computationally expensive, the PM-M method can be surprisingly fast. There are at least two reasons for it. First, we use the simplex minimization method, modified by us, which we call 'EkPMinimizer' and which is based on M. Lampton's implementation of a classic simplex minimization algorithm (Lampton, 2004). Restriction of the number of simplex points to just a few in the simplex initialization phase and using the first variant of learning with a low cross-validation fold makes the system often converge in a couple or in a dozen of iterations. The second reason for high performance of the PM-M system is its way of learning. A user is given the option of controlling in advance the number of samples retained by initializing the simplex with a different proportion of zeros and ones. Thus, for example, initializing the simplex in such a way that only about one percent of samples is retained, causes that the underlying classifier will cross-validate on approximately one percent of the samples. In the case of the second variant of learning, the classifier will be built only on a compressed dataset, which constitutes approximately one percent of the size of the current training partition and will be evaluated on the whole original training data to obtain training accuracy. This is the reason for a relatively low computational cost of the method.

The user of the PM-M system has, therefore, a lot of freedom in tuning the

input parameters to cope with both very large and very small data. Another advantage of the PM-M system over most other reference vector selection methods is the possibility of using an arbitrary underlying classifier to select prototypes, because the majority of PM-M competitors rely on the k-Nearest Neighbor rule.

The paper has the following structure. In the next section a concise explanation is given of what prototype selection methods are. Section 3 is devoted to the description of the PM-M system and the algorithms that constitute this method. In Section 4 three numerical experiments are performed. The first study is conducted on 39 small datasets not exceeding 2000 samples each. These are the same databases as those studied in one of the tests that have been conducted in Garcia et al. (2011). The IB1 (i.e. 1-Nearest Neighbor) method has been taken as the reference classifier in our comparison. Additionally, from the large number of prototype selection models tested in Garcia et al. (2011), we have taken the CHC, SSMA and GGA systems for our comparison. The latter three methods turned out to be among the best ones of many algorithms tested in the study referred to. In the experiment here reported, the majority committees of twenty PM-M models are confronted with the reference classifiers.

In the second study, which has been conducted on small datasets not exceeding 500 samples, the committees of the PM-M systems are confronted with the CHC ensembles. This calculation tests how the PM-M system performs on very small data and to what extent the committees influence generalization and stabilize the models under study.

The third numerical experiment has been conducted on one very large dataset consisting of nearly half a million samples, and was meant to show that the PM-M system can easily cope with very large data. Experiments on medium and large datasets will be more thoroughly studied elsewhere (Grudzinski, in preparation) and their analysis is out of the scope of this paper.

In this publication it is shown that the results, obtained with the PM-M system and compared with the reference ones using the corrected resampled T-Test (Hall, 2009) justify the usefulness of our PM-M method in data analysis. The last section contains conclusions that follow from the paper.

2. The prototype methods

The prototype selection methods constitute a family within a broader area, known as reference vector selection algorithms (Garcia, 2012; Jankowski and Grochowski, 2004; Grochowski and Jankowski, 2004). If n is the number of training samples, a reference vector selection method can be defined as a process of finding the smallest optimal set $\mathcal{S}$ (out of $2^n - 1$ subsets) of cases representing the same population as the original training set $\mathcal{T}$ and leading to the correct classification of the unseen cases. Quite often, a statistically indistinguishable difference in the results of classification between the prototype selection methods and the ones, in which the original unpruned training sets are taken, is obtained. Sometimes, usually for noisy data, prototype instance selection algorithms may lead to statistically significant improvement of generalization ability. The improvement often

depends on the size of the datasets under study. For very large datasets, which consist of hundreds of thousands of samples, the standard deviation of classification accuracy is often lower than one percent. In such cases, there is a higher chance for a prototype method to arrive at a statistically significant improvement of generalization over the one attained on the unpruned data. One obviously has to have a sufficiently fast prototype method at hand in order to cope with such large data.

Prototype generation (Triguero et al., 2011) extends the prototype selection framework by letting the samples change their positions. The best known example of the prototype generation algorithm is the LVQ system (Hyninen, 1996; Kohonen, 2001).

A certain number of techniques sharing the same strategies can be identified in the family of reference vector selection algorithms.

2.1. Noise filters

Noise filters or editing rules constitute a category of reference vector selection algorithms that is based on rejecting noisy cases or outliers from a training set $\mathcal{T}$. These techniques are employed as the first data preprocessing step. The rate of data pruning is usually low. After performing this step other methods come into play. ENN, RENN (Wilson, 1972), All k-NN (Tomek, 1976) and ENRBF (Jankowski, 2000) are the key examples of the algorithms belonging to this group.

2.2. Data condensation algorithms

Data condensation algorithms or data pruning (compression) techniques aim at attaining the highest possible training data reduction with minimal sacrifice of generalization. The aim of the algorithms belonging to this group is to find and to remove the training vectors that have a small influence on learning and thus their presence negatively affects classifier time requirements and memory consumption. This is usually accomplished by discarding these instances that are located far from decision borders. CNN (Hart, 1968), RNN (Gates, 1972), GA, RNGE (Bhattacharya et al., 1981), ICF (Brighton and Mellish, 2002) and Drop 1-5 (Wilson and Martinez, 1997; 2000) are among the most important systems that can be included in this group.

2.3. Prototype selection methods

This group of reference vector selection algorithms is aimed at finding an extremely low number of samples that carry particularly large amount of information and which are capable of representing a large number of training cases. The difference between data condensation algorithms and prototype selection methods is, definitely, a very subtle one. In our opinion, the prototype selection methods push the reduction of the training set to the extreme, thus allowing for slightly larger degradation of generalization ability than it is the case for the data condensation algorithms. It should not be surprising that some of the algorithms,

particularly these that permit controlling the number of samples to be retained, may be assigned either to data condensation or to the prototype selection group of reference vector selection methods. MC1 and RMHC (Skalak, 1994), IB3 (Aha, 1991), ELH, ELGrow and Explore (Cameron-Jones, 1995) and our own methods PM-M (Grudzinski, 2004) and EkP (Grudzinski, 2010) belong to the prototype selection group of methods.

3. The PM-M system

As it has been mentioned in the introduction, the general idea behind the PM-M system is as follows. To each of the training cases a binary weight* is to be assigned, this weight indicating whether a sample should be retained or not. Thus, if n is the number of training samples, we have the n-dimensional vector $\mathbf{x}$ of parameters x_i $(i = 1 \ldots n)$ to optimize. In order to find this optimal set of parameters, we use the modified version of a downhill simplex method for function minimization written by Lampton (Lampton, 2004). We refer to this as 'EkPMinimizer'. Typically, in order to find a minimum of a cost function $T(x_1, x_2, \ldots x_n)$, where $\{x_1, x_2, \cdots, x_n\}$ is a vector of parameters to be optimized, a simplex has to be built by initializing its $n+1$ points with the values of the cost function. When the initialization is completed, the simplex method is started and the minimum is found after a certain number of iterations. In fact, in the case of minimization with the 'EkPMinimizer' system, in order to find a 'good' minimum, usually only a few simplex points are required to be initialized. Thus, in cases of very highly dimensional minimization problems and/or very computationally time demanding cost functions, the time requirements for the simplex minimization method can be reduced significantly.

3.1. The algorithms

On the following pages the pseudocode for the main functions behind the PM-M system is given. Here, `C` denotes a completely arbitrary classifier that is built on the prototypes when they are selected. In this paper, as the `C` classifier, the IB1 (i.e. 1-NN) method has been used exclusively. `NPTS` is the input parameter, which takes values from the range [2,∞] and denotes the number of simplex points initializing the minimization procedure. In the Algorithm 1 the `Random`() function returns a random value from the range [0,1]. The variable `prototypeSetSize` is the input variable, which approximately determines how many prototypes will be selected. Further on in the text of this paper we sometimes refer to it as 'the strength of a training set compression'. For example, the value of 0.01 will leave approximately 1% of the samples in partial memory and the value of 0.5 will prune the training set to approximately half of its size. Another input parameter

*Because of technical reasons, the weights used in the computer implementation of the algorithms, presented in this paper, are real numbers, not binary ones, and they are casted from the floating to integer type when it is needed. It is, however, easier to explain the idea of the algorithm when these weights are treated as if they were binary.

Algorithm 1 The simplex initialization function. This algorithm has the complexity of $O(NPTS * (n + O_{CF}))$, where $NPTS$ is the number of simplex points, n is the number of training samples and O_{CF} is the complexity of the cost function algorithm.

```
double best = double.MaxValue;

void initializeSimplexPM–M(int NPTS, Classifier C,
                           DataSet Train, double prototypeSetSize){

/* The NPTS  variable is the number of 'simplex
   points'. paramVector is a vector of parameters
   that are optimized. It contains weights
   indicating whether a sample should be
   retained or rejected. For example
   paramVector[5] stores the weight of the sixth
   training sample.
*/

  double paramVector[Train.numInstances()];
  double simplex[NPTS][Train.numInstances() + 1];

  for(int i = 0; i < NPTS; i = i + 1){
   for(j = 0; j < Train.numInstances(); j = j + 1){
      paramVector[j] = Random() + prototypeSetSize;
      simplex[i][j] = paramVector[j];
   }

   simplex[i][Train.numInstances()] =
   costFunction(paramVector, C, Train);
  }
}
```

(appearing only in the case of the first variant of learning) is the number of internal cross-validation folds used for learning. In the headers of the algorithm listings, the information concerning the complexity of these algorithms is provided.

3.2. Complexity analysis

Let us start our discussion by analyzing the complexity of both variants of the cost function algorithm. The first loop in the listings of the Algorithms 2 and 3, which iterates over all n training samples is common for both variants of the cost function algorithm. Thus, both algorithms will have the complexity of at least $O(n)$. Let us denote by $O_{C_{Tr}}$ and $O_{C_{Tst}}$ the complexities of of the training and testing phases of the underlying classifier C, classifying the samples in the current training partition. Thus, the complexity of both variants of the cost function algorithm takes the value $O_{CF} = O(n) + O_{C_{Tr}} + O_{C_{Tst}}$. Recall that in this paper the IB1 method has been used exclusively as the underlying classifier C. In the first variant of learning, the C classifier crossvalidates on p prototypes. The complexity of training the IB1 classifier is $O(1)$ and the complexity of testing a single sample by the IB1 method is $O(p)$, because there are p samples in partial memory. Therefore, the complexity of a testing phase of the IB1 system is $O(p^2)$. Thus, the overall complexity of the Algorithm 2 (i.e. the first variant of the cost function algorithm) is $O(n + p^2)$.

Algorithm 2 The first variant of the cost function algorithm (learning via internal cross-validation on prototype data). If the IB1 model is taken as the underlying classifier (i.e. C≡IB1) the complexity of this variant of learning takes the value $O_{CF1} = O(n + p^2)$, where p is the number of prototypes.

```
double  costFunction(float paramVector[],
                     Classifier C, DataSet Train){

  DataSet Proto = Train; //Start from the whole trainig set

  for(i = Train.numInstances() - 1; i >= 0; i = i - 1){
     if(((int) paramVector[i]) <= 0)
        deleteInstance(i, Proto);
   }

  Evaluation ev;

  for(i = 1; i <= numFoldsLearn; i = i + 1){
        train = GetTrain(Proto,i);
        test  = GetTest(Proto,i);

        BuildClassifier(train,C);
        ev.EvaluateClassifier(test,C);
  }

  if(ev.rootMeanSqErr() < best) {
     best = ev.rootMeanSqErr();
     optimalParamV = paramVector;
  }

  return best;
}
```

Algorithm 3 The second variant of the cost function algorithm (learning via testing the underlying classifier on a whole training partition and building it on prototype cases). By choosing the IB1 model as the underlying classifier (i.e. C≡IB1) the complexity of this variant of learning takes the value of $O_{CF2} = O(n + pn)$, where p is the number of prototoypes and n is the number of training samples.

```
double  costFunction(float paramVector[],
                     Classifier C, DataSet Train){

  DataSet Proto = Train; //Start from a whole trainig set

  for(i = Train.numInstances() - 1; i >= 0; i = i - 1){
     if(((int) paramVector[i]) <= 0)
        deleteInstance(i, Proto);
  }

  Evaluation ev;
  BuildClassifier(Proto,C);
  ev.EvaluateClassifier(Train,C);

  if(ev.rootMeanSqErr() < best){
     best = ev.rootMeanSqErr();
     optimalParamV = paramVector;
  }

  return best;
}
```

In the second variant of learning, the C classifier is trained on p prototypes and is evaluated on the entire training partition, instead of crossvalidating on the training partition consisting of p prototypes, as this was the case of Algorithm 2. An analysis similar to the one above, conducted for the case of the second variant of the cost function algorithm will lead us to the conclusion that the complexity of the Algorithm 3 is $O(n + p * n)$.

Taking into account the number of attributes d in a problem domain, the complexities of both variants of the cost function algorithm take the forms $O(n + d * p^2)$ and $O(n + d * p * n)$, respectively.

In the same way, the complexity of the Algorithm 1 (i.e. the simplex initialization algorithm) can be analyzed. The complexity of this algorithm takes the value $O(NPTS * (n + O_{CF}))$.

Thus, if we restrict ourselves to the slower variant of the cost function algorithm (i.e. Algorithm 3), the complexity of the initialization of the simplex method is $O(NPTS * n + NPTS * p * n)$, which means, in fact, $O(NPTS * p * n)$. Because in real applications $NPTS \ll n$ and $p \ll n$, this complexity is very promising.

Application of the PM-M system to very large datasets shows that the entire minimization process can often be completed in less than 100 cost function calls. This proves that the method under study is easily capable of analyzing very large datasets (see Section 4.3).

4. Numerical experiments

In order to show the usefulness of the PM-M system, three numerical experiments have been conducted in which the SuperWeka system has been used (Grudzinski, 2006-2017). Thirty nine small databases (i.e. the ones that consist of no more than two thousand samples each) from the UCI repository (Lichman, 2013) and Keel repository (Alcalá-Fdez, 2011) have been taken for our first study (Table 1). The second experiment has been performed on 24 very small datasets that remained by rejecting these domains, used in the first study, which had more than 500 samples. The third experiment has been conducted on only one domain (kddcup) consisting of about half a million examples.

In the first experiment, we have employed majority committees (Kittler et al, 1998; Bauer and Kohavi, 1999; Kuncheva, 2004;) of twenty homogeneous (i.e. all with the same configuration) PM-M models, all of which have been trained with the first variant of learning. The IB1 (i.e. 1-Nearest Neighbor) system in all three experiments has been used as the reference classifier, as well as the underlying method for the PM-M systems with which the prototype cases were found. Additionally, the CHC, SSMA and GGA (Garcia, 2011) prototype selection methods have also been taken for comparison in our first numerical study[†].

[†]The latter three prototype methods are part of the Keel (Alcalá-Fdez, 2009) software, and they have been brought to the SuperWeka system by the author of this paper.

The aim of the second experiment is to show that even though the 'raw' PM-M system is slightly unstable when applied to very small datasets, its majority committees can very well compete with the majority committees of the best reference classifier taken for comparison, i.e. the CHC system. In this experiment, the CHC and PM-M majority committees have been constructed from 20 models all having the same configuration. The PM-M system has been trained this time with the second variant of learning and the number of simplex points has been set to be equal to the number of training cases.[‡] There are two reasons for the choice of only very small datasets, consisting of no more than 500 samples, in this experiment. First, the datasets have to be small enough in order to be able to complete the calculations in reasonable time. This concerns primarily the CHC model, which is very time consuming. Moreover, the second variant of learning, with which the calculations for the PM-M system have been conducted in this study, is much more time consuming than the first version of learning and therefore only small datasets were used. The second reason for the selection of only small datasets is, obviously, the need to observe to what extent the committees stabilize both methods used for the experimentation and how they influence generalization.

The 10-fold stratified cross validation test has been performed in all our experiments. The calculations have been repeated ten times in the first and the second study and the percentage of classification accuracy and the prototype set size averaged over hundred partial results have been recorded. In order to estimate the statistical significance of the difference between the obtained results, the corrected resampled T-Test with the significance factor 0.05 has been employed (Hall, 2009).

In our third calculation, carried out on the kddcup data, the 10-fold cross-validation test without repetition has been preformed. This study is a preliminary experiment, whose aim is to show that the PM-M system can easily cope with very large data by applying this method directly.

4.1. Discussion of the results of the first experiment

The numerical results of the first experiment are summarized in Tables 2–4. CHC, SSMA and GGA – the PM-M competitors – performed extremely well in our study. CHC has compressed the size of the training sets to the average value of 2.7% of their original size. This method outpaced the IB1 classifier on ten datasets, on the remaining seventeen databases the statistically indistinguishable results of classification accuracy have been obtained. Thus, only on twelve domains the degradation of generalization has occurred. SSMA and GGA have given similar classification accuracy on test sets as CHC, but these models required slightly more of the samples (4.25% and 5.85%, respectively).

Comparing the results of classification obtained with the latter two models

[‡]The detailed comparison of both variants of learning and the influence of each of the PM-M's input parameters on generalization and time requirements of the PM-M system will be given in Grudzinski (in preparation).

Table 1. Characteristics of the datasets used in our experiment

#	Dataset	Ex.	Atts.	Num.	Nom.	Cl.
1	appendicit.	106	7	7	0	2
2	australian	690	14	8	6	2
3	automobile	205	25	15	10	6
4	balance	625	4	4	0	3
5	bands	539	19	19	0	2
6	breast	286	9	0	9	2
7	bupa	345	6	6	0	2
8	car	1728	6	0	6	4
9	cleveland	303	13	13	0	5
10	contrace.	1473	9	9	0	3
11	crx	690	15	6	9	2
12	dermatol.	366	34	34	0	6
13	ecoli	336	7	7	0	8
14	flare-solar	1066	11	0	11	6
15	german	1000	20	7	13	2
16	glass	214	9	9	0	7
17	haberman	306	3	3	0	2
18	hayes-roth	160	4	4	0	3
19	heart	270	13	13	0	2
20	hepatitis	155	19	19	0	2
21	housevotes	435	16	0	16	2
22	iris	150	4	4	0	3
23	led7digit	500	7	7	0	10
24	lymphogr.	148	18	3	15	4
25	mammogr.	961	5	5	0	2
26	monk-2	432	6	6	0	2
27	newthyr.	215	5	5	0	3
28	pima	768	8	8	0	2
29	sahart	462	9	8	1	2
30	sonar	208	60	60	0	2
31	spectheart	267	44	44	0	2
32	tae	151	5	5	0	3
33	tic-tac-toe	958	9	0	9	2
34	vehicle	846	18	18	0	4
35	vowel	990	13	13	0	11
36	wine	178	13	13	0	3
37	wisconsin	699	9	9	0	2
38	yeast	1484	8	8	0	10
39	zoo	101	16	0	16	7
	Average	533.74	13.33	10.23	3.10	3.77

Table 2. Summary of the results of classification accuracy (%) of 39 small datasets obtained with the CHC, SSMA and GGA prototype selection systems. The IB1 method (i.e. 1-NN) which was trained on the whole training partitions has been taken as the reference classifier. The pairwise comparison of the results has been done using the corrected resampled T-Test (Hall, 2009) with the significance factor $\sigma = 0.05$

#	Dataset	IB1	std dev	CHC	std. dev.	size	(b/ t/ w)	SSMA	std. dev.	size	(b/ t/ w)	GGA	std. dev.	size	(b/ t/ w)
1	appendicit.	80.96	9.30	85.06	9.41	2.89		84.75	8.91	3.61		85.52	8.80	3.55	
2	australian	80.87	4.11	83.38	5.59	0.79		82.36	5.07	1.49		82.72	4.31	3.16	
3	automobile	74.40	10.27	46.10	10.67	4.81	*	53.54	10.70	9.26	*	53.03	11.34	10.48	*
4	balance	78.29	4.24	87.87	3.19	1.07	v	87.71	2.78	2.51	v	86.80	2.97	4.49	v
5	bands	62.78	6.23	59.24	5.61	1.82		60.13	6.45	4.23		59.33	6.28	6.07	
6	breast	67.89	7.52	72.67	6.91	1.64		71.20	7.73	2.28		69.24	8.17	2.85	
7	bupa	62.92	8.19	62.12	7.91	2.66		63.69	7.71	5.34		64.12	8.78	6.84	
8	car	77.30	3.18	67.46	4.93	1.08	*	69.42	4.37	3.01	*	74.71	2.97	8.39	
9	cleveland	52.47	7.41	58.42	5.57	1.43	v	56.49	5.79	2.28		54.78	6.60	3.80	
10	contrace.	42.56	3.21	49.73	4.34	0.79	v	48.32	4.19	3.24	v	47.16	4.84	8.07	v
11	crx	82.78	4.37	77.55	6.98	0.87	*	76.13	7.67	1.51	*	78.46	4.58	3.35	*
12	dermatol.	95.60	3.16	92.87	4.45	2.46		92.97	4.02	3.41		93.08	3.89	3.73	
13	ecoli	79.61	6.18	79.31	6.96	3.45		80.45	6.63	5.05		79.64	6.68	6.20	

continued in next page

Table 2 continued

#	Dataset	IB1	std. dev	CHC	std. dev.	size	(b/ t/ w)	SSMA	std. dev.	size	(b/ t/ w)	GGA	std. dev.	size	(b/ t/ w)
14	flare-solar	60.36	6.70	66.93	5.78	0.55	v	66.33	4.59	0.56	v	66.80	4.58	2.64	v
15	german	72.24	3.59	68.72	4.49	0.92		69.40	3.97	3.15		69.55	3.79	6.39	
16	glass	72.01	9.32	62.88	10.26	5.03	*	64.26	9.63	7.15		64.94	8.63	8.11	
17	haberman	65.75	7.40	73.76	5.50	1.47	v	73.89	5.61	1.90	v	74.21	5.77	2.94	v
18	hayes-roth	75.47	10.49	49.96	12.68	5.09	*	71.08	13.04	9.55		52.40	14.76	8.06	*
19	heart	76.15	8.48	82.04	6.75	1.56	v	79.78	7.43	2.65		80.59	7.99	3.53	
20	hepatitis	81.40	8.55	81.41	10.04	3.56		80.96	9.70	4.34		82.55	8.18	5.23	
21	housevotes	92.23	3.95	91.98	4.67	1.18		91.32	4.60	1.51		92.18	4.72	2.33	
22	iris	95.20	4.75	91.20	7.92	3.19		90.87	8.08	3.93		91.60	8.02	4.07	
23	led7digit	64.04	8.34	64.66	7.45	3.44		75.84	6.64	3.11	v	64.54	7.70	4.66	
24	lymphogr.	81.72	10.40	79.31	9.75	3.83		75.99	10.70	5.73		73.33	10.64	7.39	*
25	mammogr.	74.07	3.80	80.08	6.05	0.63	v	80.94	4.79	1.09	v	79.77	3.64	3.98	v
26	monk-2	76.68	6.89	96.69	3.49	0.94	v	95.95	3.57	2.45	v	88.65	5.20	6.38	v

Continued in next page

Table 2 continued

#	Dataset	IB1	std. dev	CHC	std. dev.	size	(b/ t/ w)	SSMA	std. dev.	size	(b/ t/ w)	GGA	std. dev.	size	(b/ t/ w)
27	newthyr.	96.93	4.16	91.50	6.05	2.34	*	92.55	5.96	2.81		90.74	7.48	3.20	*
28	pima	70.40	4.50	73.35	5.82	1.18		72.78	4.08	2.39		72.08	4.46	4.95	
29	sahart	65.15	6.47	70.52	6.47	1.42		69.92	6.40	2.71		69.88	6.94	3.38	
30	sonar	85.98	8.62	74.49	10.86	3.85	*	74.38	9.95	8.01	*	76.54	8.77	10.25	*
31	spectheart	69.59	6.93	77.92	5.60	1.65	v	77.13	6.65	2.56	v	76.07	6.97	3.62	v
32	tae	63.28	12.73	51.01	11.36	4.19	*	51.10	12.63	7.28	*	51.34	11.17	8.00	*
33	tic-tac-toe	80.71	4.00	67.84	4.41	1.56	*	73.97	4.58	5.32	*	77.59	4.41	8.58	
34	vehicle	69.59	3.77	63.12	5.02	2.56	*	65.82	4.75	6.45		65.52	4.50	8.58	*
35	vowel	99.04	1.07	72.79	5.34	16.40	*	87.20	4.34	15.34	*	77.41	4.93	17.08	*
36	wine	95.22	4.63	94.49	5.19	2.81		94.72	5.30	3.55		94.49	4.91	4.02	
37	wisconsin	95.42	2.24	95.95	3.13	0.55		95.88	4.24	0.69		96.15	3.21	0.97	
38	yeast	52.47	4.17	56.79	4.15	1.20	v	57.23	4.10	3.13	v	55.27	4.43	7.75	
39	zoo	96.32	5.42	91.97	5.96	8.59	*	95.92	6.19	11.30		93.72	6.76	10.99	
	Average	**76.05**		**74.18**		**2.70**	**(10/ 17/ 12)**	**75.70**		**4.25**	**(9/ 23/ 7)**	**74.53**		**5.85**	**(7/ 23/ 9)**

(v/ /*) - Statistically significant improvement or degradation, $\sigma = 0.05$

against the baseline system, we conclude that SSMA performed better than the IB1 classifier on nine datasets, whilst GGA won seven times. At the same time, only on seven datasets SSMA performed worse than the IB1 system. In the case of GGA model, only nine times out of thirty nine the degradation of the classification accuracy has been observed. The results of classification obtained with these three prototype methods are summarized in Table 2. In the columns, which are labeled by (b/t/w), this label standing for (better/ties/worse), the information is provided about whether the relevant prototype method has been better, worse or gives the same results as the baseline classifier, with respect to the given dataset. The 'v' mark stands for improvement, the asterisk '*' denotes statistically significant degradation of the classification accuracy and the lack of any mark denotes the ties.

Similar results to these obtained with the CHC, SSMA and GGA method have been attained with the first majority ensemble of twenty homogeneous PM-M systems. By the term 'homogeneous' we understand that all of the PM-M models, comprising the committee, have the same configuration. In this particular case we have chosen for the PM-M models the following values of the input parameters: the first variant of learning through 3-fold cross-validation, only two simplex points, on which 'simplex' has been built, and the initial value of 0.0585 (i.e. 5.85%) for the parameter `prototypeSetSize`, controlling the strength of the training set compression.

As it can be seen from Table 3, the results obtained with the first majority committee lead to similar generalization as in the case of calculations conducted with the CHC, SSMA and GGA models. The same holds for the training data compression. The obtained average level of data pruning of 5.95% is not much different from the value, with which each of the PM-M models started, i.e. 5.85%. This is good, because it means that the target size of the prototype memory can be estimated in advance and very precisely controlled.

What concerns the time requirements of our proposed method, the calculations required to complete the test by the first majority ensemble lasted more than roughly two orders of magnitude shorter than in the case of its prototype competitors (Table 4). Dividing this value by twenty, which is the number of models comprising the ensemble, leads to the decrement by more than three orders of magnitude concerning the time requirements of the 'raw' PM-M model with respect to the reference prototype systems studied. The results of generalization, obtained with the second ensemble, taken for our experimentation, are even better. In this case, the average value of classification accuracy is higher than the one attained with IB1 by about 1.5% and the number of datasets, on which the ensemble performed worse than the baseline system, drops to only two. Despite setting the data compression parameter to the value of thirty percent, the resulting committee has still required less time to complete the test by more than one order of magnitude than its prototype competitors.

Table 3. Summary of the results of classification accuracy (%) of 39 small datasets obtained with the two majority voting ensembles composed of 20 homogeneous IB1-based PM-M systems and with the IB1 method, trained on the whole training partitions, used as a baseline classifier. The number of internal cross-validation learning folds, denoted by the -X flag, was set to 3. The number of used simplex points was first set to 2 and then to 10% of the training set size (flag -N). The pairwise comparison of the results was done using the corrected resampled T-Test (Hall, 2009) with the significance factor $\sigma = 0.05$

#	Dataset	IB1	std. dev.	Vote −X = 3 −N = 2 20	std. dev.	initial size 5.85%	(b/ t/ w)	Vote −X = 3 −N = 10% 20	std. dev.	initial size 30%	(b/ t/ w)
1	appendicitis	80.96	9.30	82.48	5.31	5.85		86.90	7.56	29.97	
2	australian	80.87	4.11	86.01	3.61	5.89	v	84.71	3.67	29.98	v
3	automobile	74.40	10.27	52.29	9.58	5.95	*	69.19	10.07	30.28	
4	balance	78.29	4.24	87.79	2.50	5.86	v	83.74	3.31	30.04	v
5	bands	62.78	6.23	58.98	5.84	5.86		62.87	6.19	29.92	
6	breast	67.89	7.52	72.37	4.55	5.83		72.48	6.05	30.16	
7	bupa	62.92	8.19	61.56	7.14	5.86		62.73	8.06	30.00	
8	car	77.30	3.18	85.20	2.74	5.86	v	86.34	2.37	30.04	v
9	cleveland	52.47	7.41	57.86	4.65	5.91		56.50	7.47	29.92	v
10	contraceptive	42.56	3.21	48.20	4.06	5.87	v	45.55	3.53	30.02	v
11	crx	82.78	4.37	85.87	4.22	5.87	v	86.22	4.11	30.00	v
12	dermatology	95.60	3.16	95.14	3.60	5.89		96.48	2.65	30.00	
13	ecoli	79.61	6.18	74.12	6.65	5.87	*	81.76	6.02	30.04	
14	flare-solar	60.36	6.70	64.95	4.19	5.82		63.04	5.61	30.03	
15	german	72.24	3.59	72.46	2.95	5.88		73.05	3.54	29.99	
16	glass	72.01	9.32	61.65	8.84	6.06	*	70.05	8.79	30.01	
17	haberman	65.75	7.40	74.03	4.51	5.86	v	69.64	5.80	30.02	
18	hayes-roth	75.47	10.49	45.73	11.68	5.88	*	57.88	14.49	30.33	*
19	heart	76.15	8.48	82.00	6.78	5.89	v	79.81	7.03	29.96	
20	hepatitis	81.40	8.55	80.82	5.10	5.88		83.06	8.69	30.14	

Continued in next page

Table 3 continued

#	Dataset	IB1	std. dev.	Vote −X = 3 −N = 2 20	std. dev.	initial size 5.85%	(b/ t/ w)	Vote −X = 3 −N = 10% 20	std. dev.	initial size 30%	(b/ t/ w)
21	housevotes	92.23	3.95	91.70	3.97	5.86		92.95	3.64	30.01	
22	iris	95.20	4.75	93.93	5.85	6.14		95.27	5.04	29.92	
23	led7digit	64.04	8.34	73.66	6.63	5.91	v	71.34	6.74	30.05	v
24	lymphography	81.72	10.40	79.56	9.56	6.01		84.17	8.09	30.13	
25	mammographic	74.07	3.80	81.14	3.71	5.87	v	78.69	3.80	30.09	v
26	monk-2	76.68	6.89	81.53	6.86	5.92		89.45	4.99	30.00	v
27	newthyroid	96.93	4.16	87.22	5.86	5.95	*	94.61	4.37	29.96	
28	pima	70.40	4.50	73.32	4.26	5.86		73.08	4.53	30.00	v
29	sahart	65.15	6.47	70.80	5.64	5.84	v	67.70	5.63	30.03	
30	sonar	85.98	8.62	72.01	8.95	5.86	*	84.57	8.10	30.09	
31	spectheart	69.59	6.93	79.11	5.80	5.87	v	73.54	6.81	29.97	
32	tae	63.28	12.73	50.11	12.38	5.94	*	56.43	11.68	30.16	
33	tic-tac-toe	80.71	4.00	87.17	2.97	5.89	v	89.86	3.64	29.96	v
34	vehicle	69.59	3.77	68.39	4.38	5.87		70.80	4.27	30.00	
35	vowel	99.04	1.07	62.14	5.07	5.88	*	96.19	2.10	30.07	*
36	wine	95.22	4.63	95.96	4.67	6.18		96.41	4.41	30.11	
37	wisconsin	95.42	2.24	96.22	2.44	5.80		96.58	2.20	29.99	
38	yeast	52.47	4.17	57.94	3.88	5.86	v	56.70	3.83	30.00	v
39	zoo	96.32	5.42	83.66	9.07	8.09	*	94.14	6.35	30.17	
	Average	**76.05**		**74.75**		**real size 5.95%**	**(13/ 17/ 9)**	**77.81**		**real size 30.04%**	**(12/ 25/ 2)**

(v/ /*) - Statistically significant improvement or degradation, $\sigma = 0.05$

Table 4. The program average user execution times (seconds) obtained by running each of the models that has been used in our experiments on a single cross validation fold. Tr stands for average user training time, Tst - average user testing time, and Tot is the user total time. The -P input option denotes the strength of training data compression. The -X input option denotes the internal cross-validation learning fold with which the first variant of the cost function is executed. Finally, the -N input variable denotes the number of 'simplex points' taken for the calculation. The values with the % mark denote the used options expressed in terms of percentage of the training set size

	IB1	**CHC**	**SSMA**	**GGA**	**Vote 1**	**Vote 2**
Tr	0.0417	16.9790	9.9369	16.5900	0.0169	0.4979
Tst	0.0037	0.0001	0.0002	0.0004	0.0063	0.0276
Tot	0.0454	16.9791	9.9371	16.5904	0.0232	0.5255
	Vote 1 Settings:	-P 5.85%	-X 3	-N 2	20 models	
	Vote 2 Settings:	-P 30%	-X 3	-N 10%	20 models	

4.2. Discussion of the results of the second experiment

In the second study, the majority committees, composed of twenty PM-M models, have been confronted with twenty model-based committees of the CHC systems. The experiments in this section have been conducted on 24 small datasets that have been filtered out from the ones listed in Table 1 by rejecting these domains, which had more than 500 cases. Let us first discuss the results obtained with the 'raw' (i.e. not committee) models. The PM-M competitor, the CHC system, performed extremely well in this study. In the 10-fold cross-validation test, which has been repeated ten times, the 'raw' CHC system attained 75.93% of classification accuracy on unseen samples. At the same time, with the IB1 system, which was trained on the whole original training partitions, 77.79% of classification accuracy has been obtained. As it is usually the case of the CHC system, a very large training set compression ratio has been recorded in this calculation. The size of the outcoming prototype sets had the size, on the average, of only 3.06% of the original training sets. In order to perform a fair comparison between the 'raw' CHC model and the 'raw' PM-M system, the latter should be evaluated in the same way as its competitor. Additionally, one has to assure that the PM-M selects the prototype sets of approximately the same size as CHC. This can be done with PM-M by initializing the simplex with the appropriate proportion of 0's and 1's. Despite choosing for the PM-M system this time the settings that assure its best performance, i.e. the second variant of learning and initializing all the simplex points, in this test the CHC system outperformed the raw PM-M model, with which only 63.15% of classification accuracy has been obtained. The average size of the dataset compressed by PM-M has been equal to 3.91%. This result indicates that the calculations with the 'raw' PM-M model lead to poor generalization when this method is trained on not sufficiently large learning sets and, additionally, a very low number of prototypes is expected to be obtained.

Things look completely different when committees of PM-M models are used. The results of the second experiment are summarized in Table 5. The role of the baseline classifier is now played by the majority committee of twenty CHC models. All the PM-M models, comprising the committee, have been used with the same configuration, the same as the one with which the raw PM-M system has been used (namely with the second variant of learning and with the initialization of all simplex points). As it has been already mentioned, the raw IB1 system attained in this study 77.79% of classification accuracy. The committee of twenty PM-M systems has been very competitive with respect to the CHC ensemble. PM-M outperformed CHC on one dataset out of 24, and at the same time CHC won twice on two different domains. For the remaining databases, the statistically indistinguishable differences of the results of classification have been obtained. What concerns the user training time of the models under study, the PM-M committee outpaced the CHC ensemble about four times, and the calculations with these two models lasted, on the average, 4.37 and 16.92 seconds, respectively, on a single cross-validation fold. Majority committees very much stabilized the results obtained with the PM-M system. The average standard deviation of classification accuracy which was taken over all 24 datasets and which has been obtained by the PM-M ensemble, was equal to 6.63%. At the same time, the value of 11.12% has been obtained when the raw PM-M system has been taken for the calculation. The influence of using the committee on stabilization of the CHC system was marginal, with about 1% decrement of the average standard deviation of classification accuracy with respect to the value, which was obtained with the raw CHC model.

As the results illustrate this on the small datasets, the majority voting makes the PM-M system competitive with respect to the best prototype classifiers in the field. This holds not only in the case of classification accuracy, but also in the case of size of prototype sets. There is, however, a problem with the interpretation of importance of the selected prototypes. In the case of the majority voting, the information about the significance of the particular prototypes for classification is scattered in committee models. This issue requires further examination. One of solutions for this problem seems to be the application of prototype based committees (Grudzinski, 2006). With the help of the prototype based committees there will probably be no need to look for other prototype selection methods than PM-M when analyzing also very small datasets. Although the prototype based committees have been used with good results earlier, their utility needs to be experimentally verified again on a larger scale.

4.3. Suitability for very large data analysis

The study of suitability of the PM-M system for medium and very large data analysis is out of the scope of this paper. Classification of medium and large datasets as well as the other PM-M properties will be studied in Grudzinski (in preparation). Therefore, for purposes of this paper the experiment on only one very large dataset has been performed. This particular experiment was meant to

Table 5. Summary of results of classification accuracy (%) of 24 small datasets (i.e. not exceeding 500 samples) obtained with the IB1 method and majority voting ensembles of PM-M and CHC prototype selection systems. The CHC committee has been taken as the reference (baseline) classifier. The pairwise comparison of the results has been done using the corrected resampled T-Test (Hall, 2009) with the significance factor $\sigma = 0.05$.

#	Dataset	Vote CHC 20	std. dev.	size	IB1	std. dev.	(b/ t/ w)	Vote PM-M 20	std. dev.	size	(b/ t/ w)
1	appendicit.	85.15	8.58	2.87	80.96	9.32		86.88	8.61	4.31	
2	automobile	54.27	8.41	4.91	74.40	10.29	v	55.53	7.58	4.39	
3	breast	75.11	6.55	1.68	67.89	7.52	*	74.10	5.66	3.21	
4	bupa	65.79	7.71	2.65	62.92	8.19		64.24	7.12	3.58	
5	cleveland	59.60	4.97	1.45	52.47	7.41	*	58.08	4.20	3.34	
6	dermatol.	95.57	3.73	2.50	95.60	3.16		96.39	2.96	4.18	
7	ecoli	81.97	6.30	3.52	79.61	6.18		77.44	6.16	4.05	*
8	glass	68.56	8.81	5.04	72.01	9.32		62.98	8.58	4.50	
9	haberman	74.31	5.92	1.50	65.75	7.40	*	75.10	4.87	3.12	
10	hayes-roth	52.51	13.71	5.11	75.47	10.49	v	56.94	14.02	4.43	
11	heart	83.37	7.03	1.67	76.15	8.48	*	82.41	6.19	3.25	
12	hepatitis	82.72	8.17	3.48	81.40	8.55		84.77	7.89	4.06	
13	housevotes	94.57	3.06	1.19	92.23	3.95		94.51	3.34	3.27	
14	iris	93.93	6.15	3.22	95.20	4.75		94.93	5.62	4.42	
15	ed7digit	71.90	6.21	3.42	64.04	8.34	*	75.14	6.43	4.04	v
16	lymphogr.	83.80	8.27	3.91	81.72	10.40		81.46	8.54	3.97	
17	monk-2	97.22	2.66	0.97	76.68	6.89	*	87.64	4.93	3.32	*
18	newthyr.	94.47	4.26	2.34	96.93	4.16		94.22	4.46	3.79	
19	sahart	71.03	6.46	1.37	65.15	6.47	*	73.22	5.98	3.04	
20	sonar	77.65	9.07	4.00	85.98	8.62	v	75.13	8.72	3.96	
21	spectheart	80.24	4.23	1.61	69.59	6.93	*	80.43	3.79	3.58	
22	tae	53.20	11.26	4.38	63.28	12.73		51.80	12.73	4.59	
23	wine	96.90	3.81	2.82	95.22	4.63		97.36	3.52	4.02	
24	zoo	92.45	6.20	8.55	96.32	5.42	v	89.71	7.15	7.75	
	Average	78.60		3.09	77.79		(4/ 12/ 8)	77.93		4.01	(1/ 21/ 2)
(v/ /*) - Statistically significant improvement or degradation, $\sigma = 0.05$											

convince the reader that the PM-M system is, as well, a very good tool in large data analysis.

Direct application of most of the known prototype selection methods to large data is usually avoided because of the computational cost involved. Instead, a stratification procedure can be performed (Cano, 2005). The PM-M system can cope with large data directly, provided that the outcoming target prototype set is sufficiently small and a very low number of simplex points is chosen.

We have taken the kddcup dataset consisting of 424,020 instances, 42 attributes (including the class attribute) and 23 classes for our experimentation. Because for such a large dataset the standard deviation of classification accuracy is low, only one repetition of the 10-fold cross-validation test with the IB1 and PM-M classifier has been performed. It took 0.16 second of user training and about 121 minutes of user testing time for the IB1 model to complete the calculation on a single cross-validation fold. The average testing classification accuracy (the average was taken over all 10 folds) of 99.96% have been obtained. At the same time, the PM-M system completed the classification of all the samples from a single cross-validation fold on the average in 38.22 seconds. It took only 8.17 seconds of the user training time to train this classifier and the testing phase lasted for 30.05 seconds of the user testing time. Classification accuracy of 99.49% has been attained with the PM-M system on unseen samples. What concerns the values of the input parameters that have been set for the PM-M method, the first variant of learning has been used by performing 3-fold internal cross-validation. The value of the strength of the instance compression parameter has been set to 0.005 and 50 simplex points have been taken for the calculation. It took on the average only 101 cost function calls for the PM-M system to converge to a minimum root mean squared classification error on a single cross-validation fold. The training set partitions have been compressed by the PM-M model on the average to 0.5% of their original size. Taking more simplex points and increasing the value of the compression parameter would probably result in attaining better generalization by the PM-M system, but this requires empirical verification. Thus, further experiments, aiming at investigation of the influence of each of the input parameters on generalization ability and on the time requirements of the method need to be conducted and their results will be reported in the subsequent publications (Grudzinski, in preparation).

5. Conclusions and further research

In this paper, the algorithm for the PM-M system has been provided. Numerical experiments on small datasets not exceeding 2000 samples and on one very large dataset (424,020 instances) indicate that the system under study is a competitive prototype selection method and that it allows to reduce substantially the training set size, while keeping classification accuracy at a statistically good level. However, classification accuracy, although definitely very important, is not the only measure of the suitability of the method. The advantage of the proposed system over most of other reference selection algorithms is the possibility of selection of

the target prototype sets, which are of the size that can be approximately estimated and controlled in advance. Thus, the PM-M system may be used to prepare the training data of arbitrary size for other slow algorithms that could normally not cope with large data. Finally, a strong side of the proposed algorithm is its speed. With this respect, the PM-M system has the advantage over many other methods.

What requires further study is the influence of the adopted number of simplex points on generalization ability of the method and on the time requirements of this method. The same concerns the choice of the learning variant, with which the PM-M method is trained. A larger scale comparison with other well known reference selection methods has to be conducted and sensitivity to noise has to be examined. Optimal values for the number of selected prototypes for individual datasets have to be found and after that, the computations have to be repeated. More calculations on very large datasets have to be performed in order to show that the method can cope with large data. Thus, further experiments aiming at investigation of the mentioned issues will be conducted and the results will be presented in the subsequent publications (Grudzinski, in preparation).

Acknowledgments

I would like to thank my colleagues, R. Adamczak, M. Blachnik and M. Grochowski, for reading the draft of this paper and for very valuable remarks that helped in preparation of the final version of this article.

References

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

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell J. M., Otero, J., Romero, C., Bacardit, J., Rivas, V. M., Fernández, J. C., Herrera, F. (2009) KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems. *Soft Computing* **13**:3 307-31.

Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F. (2011) KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing* **17**:2-3, 255-287.

Bauer, E., Kohavi, R. (1999) An empirical comparison of voting classification algorithms:bagging, boosting and variants. *Machine Learning* 36, 105-142.

Bhattacharya, B., Poulsen, R. and Toussaint, G. (1981) Application of proximity graphs to editing nearest neighbor decision rule. In: *IEEE International Symposium on Information Theory*, Santa Monica. IEEE.

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

Cameron-Jones, R. (1995) Instance selection by encoding length heurystic with random mutation hill climbing. In: *Xin Yao, ed., Proceedings of the Eighth Australian Conference on Artificial Intelligence*, Canberra. World Scientific Publishing, River Edge, 99–106.

Cano, J. R., Herrera, F. and Lozano, M. (2005) Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters*, **26**, 7, 953–963.

Duch, W., Blachnik, M. (2004) Fuzzy rule-based systems derived from similarity to prototypes. *Lecture Notes in Computer Science*, **3316**, 912-917.

Duch, W., Grudziński, K. (2001) Prototype based rules - new way to understand the data. *IEEE International Joint Conference on Neural Networks*, Washington D.C., 1858–1863.

Garca, S., Derrac, J., Cano, J. R. and Herrera, F. (2012) Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34**:3 417–435.

Gates, G. (1972) The reduced nearest neighbor rule. *IEEE Transactions on Information Theory* 18, 665–669.

Grochowski, M. and Jankowski, N. (2004) Comparison of Instances Selection Algorithms II: Results and Comments. *Lecture Notes in Artificial Intelligence*, **LNAI 3070**, 580–585 .

Grudziński, K. (2004) SBL-PM-M: A System for Partial Memory Learning. *Lecture Notes in Artificial Intelligence*, **LNAI 3070**, 586–591.

Grudziński, K. (2006) Prototype-Based-Committees. In: A. Cader, L. Rutkowski, R. Tadeusiewicz, J. Zurada, eds., *Artificial Intelligence and Soft Computing*. Academic Pblishing House Exit, Warsaw, 237-244.

Grudziński, K. (no date) SuperWeka: A modified Weka version developed by Karol Grudziński.

Grudziński, K. (2010) Selection of Prototypes with the E*k*P System. *Control and Cybernetics*, **39**, 2, 487–503.

Grudziński, K. (no date) Further Experiments with the E*k*P and PM-M Prototype Selection Systems. In preparation.

Grudziński, K. and Duch, W. (1996-2017) SBL: Similarity Based Learner: Software developed by Karol Grudziński and Włodzisław Duch.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten I. H. (2009) The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, **11**, 1.

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

Jankowski, N. (2000) Data regularization. In: L. Rutkowski, R. Tadeusiewicz, eds., *Proc. of the Fifth Conference: Neural Networks and Soft Computing*, Zakopane, Poland, 209–214.

Jankowski, N. and Grochowski, M. (2004) Comparison of Instances Selection Algorithms I: Algorithms Survey. *Lecture Notes in Artificial Intelligence*, **LNAI 3070**, 598–603.

Kittler,J., Hatef, M., Duin, R. P. W., Matas J. (1998) On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**, 3, 226–239.

Kohonen, T., Hyninen, J., Kangas, J., Laaksonnen, J., and Torkolla, K. (1995) LVQ_PAK: The Learning Vector Quantization Program Package. Version 3.1. Helsinki University of Technology, Laboratory of Computer and Information Science.

Kohonen, T. (2001) *Self-Orgainizing Maps.* 3rd ed. Springer-Verlag, Berlin, Heidelberg.

Kuncheva,L. (2004) *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley and Sons, Inc..

Lampton M. (2004) neldermead.java: an implementation of the Nelder & Mead simplex method. www.ssl.berkeley.edu/∼mlampton/neldermead.java

Lichman, M. (no data) UCI Machine Learning Repository [`http://archive.ics.uci.edu/ml`]. Irvine, CA: University of California, School of Information and Computer Science.

Nelder, J. and Mead, R. (1965) A simplex method for function minimization. *Computer Journal* **7**, 308-313.

Skalak, D. (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: *Proceedings of the Eleventh International Conference on Machine Learning.* Morgan Kaufman, 293–301.

Tomek, I. (1976) An experiment with the edited nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* 6, 448–452.

Triguero, I., Derrac, J., Garca, S. and Herrera, F. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, **42**. 1., 86-100.

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

Wilson, D. and Martinez, T. (1997) Instance Pruning Techniques. In: D. Fisher, ed., *Machine Learning: Proceedings of the Fourteenth International Conference*. Morgan Kaufmann Publishers, San Francisco, CA., 404-417.

Wilson, D. and Martinez T. (2000) Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, 38, 257-286.