Control and Cybernetics

vol. **XX** (XXXX) No. X

Selection of Prototypes with the EkP System

by

Karol Grudziński

Institute of Physics
Kazimierz Wielki University
Bydgoszcz, Poland.

e-mail: grudzinski.k@gmail.com

Abstract: A completely new system for a selection of reference instances, which is called EkP (**E**xactly k **P**rototypes) has been introduced by us recently. In this paper we study a suitability of the EkP method for training data reduction on seventeen datasets. As the underlying classifier the well known IB1 system (1-Nearest Neighbor classifier) has been chosen. We compare generalization ability of our method to performance of IB1 trained on the entire training data and performance of LVQ for which the same number of codebooks has been chosen as the number of prototypes which has been selected by the EkP system. The results indicate, that even with only a few prototypes which have been chosen by the EkP method, on nearly all seventeen datasets statistically indistinguishable results from these attained with IB1 have been obtained. On many datasets generalization ability of the EkP system has been larger than the one attained with LVQ.

1. Introduction

Data mining is commonly employed in many domains. A case-based way of data explanation is very popular among researchers. Such an approach to knowledge discovery and understanding is particularly often employed in medicine, where a medical doctor makes a diagnosis by referring to other similar cases in a database of patients.

Interesting instance vectors, known as reference cases, can be either selected from training data or can be generated out of a training set. In the latter case instances' features have in general different values than the ones that are stored in the original training set. Both techniques (i.e. instance selection and prototype generation) often lead to a significant training set size reduction.

This paper concerns the first above mentioned problem, i.e. 'instance selection', 'training data compression, reduction or pruning'. The idea behind this machine learning paradigm is that only a small fraction of a usually much larger, original training set is used for a final classification of unseen samples (Maloof

M., Michalski, R. , 2000; Martinez T., Wilson D. , 1997, 2000; Grochowski M. , 2003; Grochowski M., Jankowski N. , 2004-1,-; Duch. W., Grudzinski. K , 2000; Grudzinski K. , 2004, 2008).

Prototype selection is an extremely important problem which has been frequently studied by machine learning and pattern recognition researchers. Selection of reference instances can significantly speed up classification and analysis of data later and usually leads to better data understanding and may lower sensitivity to noise of some classifiers. Strong training set reduction may sometimes result in statistically significant degradation of the classification accuracy attained on unseen samples, however as many experiments illustrate often it is the other way around, i.e. data pruning improves generalization ability of classifiers. Samples selected with the EkP system can be used for example to build prototype-based rules, which had been introduced by Duch et. al. (Duch W., Grudzinski K. , 2001; Blachnik M., Duch W. , 2004) and which are a very interesting alternative to classic logical rules.

The acronym EkP is short for **E**xactly-**k**-**P**rototypes. We want to stress here that our new system differs completely from our earlier model, PM-M (Grudzinski K. , 2004).

2. Methodologies for Reference Instances Selection

Before we proceed to presentation of the EkP system and the results obtained with this method, a very concise review of some of the known techniques employed in selection of the reference cases is provided. This presentation draws heavily on the excellent work of Grochowski contained in his M.Sc. thesis (Grochowski M. , 2003).

2.1. Problem Formulation

The problem of selection of the reference instances can be defined as a process of finding the smallest set $\mathcal{S}$ of cases representing the same population as the original training set $\mathcal{T}$ and leading to correct classification of the samples from not only $\mathcal{T}$ but more importantly of the unseen cases with minimal degradation of the generalization ability of the underlying classifier. In other words, reference selection is a method for selection or generation of the most informative samples from $\mathcal{T}$ and rejection of the noisy cases or of these instances that degrade the generalization when the original training set $\mathcal{T}$ is used for learning. Thus, restricting ourselves to prototype selection by which we understand selection of reference cases in which $\mathcal{S}$ is a subset of $\mathcal{T}$, the problem is to find optimal subset $\mathcal{S}$ of all possible $2^n - 1$ subsets with respect to generalization ability of the underlying classifier. By n, the number of samples of the original training set $\mathcal{T}$ is denoted.

The reference vectors selection algorithms can be divided into a few number of techniques that share the same strategies.

2.1.1. Noise Filters

This category of methods, known also as editing rules, is based on rejecting noisy cases or outlayiers from $\mathcal{T}$. The rate of data pruning is usually low and these techniques are usually employed as the first data preprocessing step which is then followed by other methods. ENN, RENN (Wilson D. , 1972), All k-NN (Tomek I. , 1976) and ENRBF (Jankowski N. , 2000) are the key examples of the algorithms that belong to this group.

2.1.2. Data Condensation Algorithms

This group of methods is also known as data pruning or data compression techniques. The main idea behind this approach is to achieve the highest possible training data reduction without or with minimum sacrification of generalization of the employed underlying classifiers. CNN (Hart P. , 1968), RNN (Gates G. , 1972), GA, RNGE (Bhattacharya B. K., Poulsen R. S., Toussaint G. T. , 1981), ICF (Brighton H., Mellish C. , 2002) and DROP 1–5 (Martinez T., Wilson D. , 2000) are the main systems that fell into this category.

2.1.3. Prototype Methods

The family of reference selection algorithms that are aimed at finding extremely low number of highly informative super–vectors, carrying particularly large amount of information and capable of representing large number of cases, are known as prototypes methods. However the difference between data condensation algorithms and prototype methods is very subtle, in our understanding prototype selection and generation algorithms push the reduction of the training data to the extreme taking sometimes the risk of slightly larger degradation of generalization of the underlying classifiers. Thus, however both groups of methods try to arrive at the smallest set $\mathcal{S}$, the stress in data pruning techniques is put on generalization, whilst in the case of prototype algorithms it is on the extremely low amount of samples that are selected. It should not be surprising, that some of the algorithms, particularly these in which one has the control over the amount of the samples selected, may be treated either as data pruning methods or as prototype selection models. LVQ (Kaski S., Kohonen T., Oja M. , 2003), MC1 and RMHC (Skalak D. , 1994), IB3 (Aha D., Albert M., Kibler D. , 1991), ELH, ELGrow and Expolore (Cameron-Jones R. , 1995) and our own models PM-M (Grudzinski K. , 2004) and EkP (Grudzinski K. , 2008) can be included into the prototype selection group of methods.

3. The EkP System

The EkP system is based on a minimization of a cost function which returns the number of errors the classifier makes. Despite of this, the EkP method is extremely fast because during every evaluation of the cost function the reduced

training set is constructed out of only the preset number of k instances. It takes seconds for the EkP method to perform 10-fold cross-validation on most common UCI datasets. In our implementation we used the well known simplex method (Nelder J., Mead R. , 1965) for function minimization which we have taken from the Internet (Lampton M. , 2004).

The simplex must be initialized first before a minimization procedure is started. The EkP system is very sensitive to the way in which the simplex is initialized and therefore we have decided to provide the EkP's initialization algorithm which is given below. We have found inclusion of this pseudocode very important for the replication of this method.

Algorithm 1 The EkP's simplex initialization algorithm

Require: A training set **trainInstances**
Require: A vector $p[]$ of optimization parameters (**numProtoPerClass** * **numClasses** * **numAttributes** dimensional)
Require: A matrix **simplex** to construct a simplex
  Let **numPoints** denote the number of points to build simplex on
  for $i = 0$ to **numPoints** - 1 **do**
    for $j = 0$ to **numClasses** * **numProtoPerClass** - 1 **do**
      for $k = 0$ to **numAttributes** - 1 **do**
        $p[k +$ **numAttributes** * $j] :=$ **trainInstances**$[i][k]$
      end for
      simplex$[k]$[**numAttributes**] := **costFunction**$(p[])$
    end for
  end for

Two variants of the cost function algorithm have been implemented in our system. The first variant is based on the internal cross-validation learning on training partitions whilst in the second algorithm variant a classifier is trained by conducting a plain test (the pruned training partitions are used for learning and the test on the entire training partition is used for estimating training accuracy). The details about both variants of the cost function algorithm are given in the pseudocode listings which are given below.

Our implementation of the EkP method is not the simplest one as our code will become a basis for an extended version of this algorithm. In order to give a short description of the algorithm in the text of the paper, it is worth mentioning that the array of optimization parameters is (numProtoPerClass * numClasses * numAttributes) dimensional but the instances stored in this vector are not involved in any parameter modification. They are simply extracted from the parameter vector and are added to the training partition in every cost function evaluation. In other words the training partitions are built by extracting samples from a parameter vector which always contains numProtoPerClass examples from every class occurring in a problem domain. In a simpler implementation one could store the indices of the training set instances instead of storing the

Algorithm 2 The E*k*P-1 cost function algorithm (learning via internal cross-validation)

Require: A training set **trainInstances**
Require: A vector p[] of optimization parameters (**numProtoPerClass** * **numClasses** * **numAttributes** dimensional)
  for $k = 1$ to **numCrossValidationLearningFolds do**
    Create the empty training set **cvTrain**
    Build the k-th test partition **cvTest**
    for $i = 0$ to **numClasses** * **numProtoPerClass** - 1 **do**
      for $j = 0$ to **numAttributes** - 1 **do**
        Add the prototype stored in p[] starting from p[j + **numAttributes** * i] and ending in p[**numAttributes - 1** + **numAttributes** * i] to **cvTrain**
      end for
    end for
    Build (train) the classifier on **cvTrain** and test it on **cvTest**
  end for
  Remember the optimal p[] value and the associated with it lowest value of **numClassificationErrors**
  return numClassificationErrors

Algorithm 3 The E*k*P-2 cost function algorithm (learning via test on the entire training partition taking pruned training partition for building (training) a classifier)

Require: A training set **trainInstances**
Require: A vector p[] of optimization parameters (**numProtoPerClass** * **numClasses** * **numAttributes** dimensional)
  Create the empty training set **tmpTrain**
  for $i = 0$ to **numClasses** * **numProtoPerClass** - 1 **do**
    for $j = 0$ to **numAttributes** - 1 **do**
      Add the prototype stored in p[] starting from p[j + **numAttributes** * i] and ending in p[**numAttributes - 1** + **numAttributes** * i] to **tmpTrain**
    end for
  end for
  Build (train) the classifier on **tmpTrain** and test it on **trainInstances**
  Remember the optimal p[] value and the associated with it lowest value of **numClassificationErrors**
  return numClassificationErrors

Table 1. Datasets used in our experiments

#	Dataset	# Instances	# Attributes	# Numeric	# Nominal	# Classes	Base Rate (%)	Rnd. Choice (%)
1	appendicitis	106	8	7	1	2	80,18	50,00
2	breast-cancer	286	10	0	10	2	70,30	50,00
3	horse-colic	368	23	7	16	2	63,05	50,00
4	credit-rating	690	16	6	10	2	55,51	50,00
5	german_credit	1000	21	8	13	2	70,00	50,00
6	pima_diabetes	768	9	8	1	2	65,11	50,00
7	glass	214	10	9	1	6	35,51	16,67
8	cleveland-heart	303	14	6	8	2	54,45	50,00
9	hungarian-heart	294	14	6	8	2	63,95	50,00
10	heart-statlog	270	14	13	1	2	55,56	50,00
11	hepatitis	155	20	2	18	2	79,38	50,00
12	labor	57	17	8	9	2	64,67	50,00
13	lymphography	148	19	0	19	4	54,76	25,00
14	primary-tumor	339	18	0	18	21	24,78	4,76
15	sonar	208	61	60	1	2	53,38	50,00
16	vote	435	17	0	17	2	61,38	50,00
17	zoo	101	18	0	18	7	40,61	14,29
Average		337,76	18,18	8,24	9,94	3,76	58,39	41,81

numProtoPerClass * numClasses vectors themselves in the parameter array. Note that numAttributes denotes the total number of attributes in a dataset including the class attribute.

4. Numerical Experiments

In order to verify suitability of the EkP system for data analysis the classification experiments on seventeen real-world problems (mainly taken from the well-known UCI repository of machine-learning databases (Mertz C., Murphy P.)) have been performed. The information about the datasets used can be found in Table 1. The EkP system can be based on an arbitrary classifier, i.e. it can be a neural-network, support-vector machine or a decision-tree method, etc. In our experiments the IB1 (Aha D., Albert M., Kibler D. , 1991) system has been used both as the underlying classifier for the EkP system and as the reference method. The reason for selecting the IB1 system is that this method requires very small training datasets which may consist of just a few samples in order to make classification possible. Other classifiers, including IBk (Aha D., Albert M., Kibler D. , 1991) require slightly larger training sets in order to operate. Our aim when we were conducting the experiments for this paper was to show that even the calculations with the extremely low number of prototypes selected may lead to attaining excellent results on unseen samples. The well known LVQ method (Hyninen, Kangas, Kohonen, Laaksonnen, Torkolla , 1996; Kohonen T. , 2001; Kaski S., Kohonen T., Oja M. , 2003), which is however a prototype-generation system, has also been taken as the reference model in our experiments. The second reason for choosing the IB1 classifier as the underlying method for the EkP system is the fact that the LVQ method uses the k-Nearest Neighbor classifier as its classification engine.

Generalization ability of the E*k*P system with only one, two and three instances per class selected from a training set has been compared to the classification performance of LVQ for which the same number of codebooks has been used. Additionally, the results obtained with the IB1 (1-Nearest Neighbor) system which has been trained on the entire cross-validation training partitions (i.e. all training samples from every learning fold have been used) are provided.

Ten-fold stratified cross-validation test has been performed for all seventeen domains. In the experiments conducted with the E*k*P system, in each cross-validation fold, the training partition has been pruned so that only the prototype cases remained, the E*k*P's underlying classifier has been trained and it's generalization ability has been estimated on the cross-validation test partition. After the completion of the calculation on all ten folds the test has been repeated ten times and the average classification accuracy and its standard deviation which were taken over the all available hundred partial results have been reported.

The single corrected re-sampled T-Test (Frank E. Witten I., , 2000; Dobosz K. , 2006) has been used to calculate statistical significance of the results (with the factor of 0.05) in order to help making the decision whether the E*k*P system performed better, the same or worse than the reference models.

The LVQWeka implementation of the LVQ method that has been employed in our calculations was written by Jason Brownlee (Brownlee J. , 2004). Finally, what remains to be mentioned is, that the E*k*P system has been written by the author in the Java programming language as a plugin to the well known Weka machine learning workbench (Frank E. Witten I., , 2000).

4.1. Experiment 1: Generalization Ability – E*k*P vs. IB1

In the first experiment our system under study has been compared to the performance of IB1 on all seventeen domains. The results of the statistical tests against the majority classifier, both of IB1 and E*k*P, have not been contained in our paper. The base rate results however, which are the values obtained by the majority classifier[1] on all tested datasets are listed in Table 1. It is worth mentioning that IB1 appeared to outperform the majority classifier on thirteen domains. On appendicits, breast-cancer, german-credit and hepatitis datasets the results have been statistically insignificant.

The E*k*P system has been used mainly with the same default settings for all seventeen problems because the calculations have been performed in a batch mode which made performing numerical experiments and collecting the results for the paper much easier. The simplex cost function tolerance has been set to 1E-16 and the maximum number of cost function evaluations has been restricted to 300 calls excluding a certain number of target function evaluations required to initialize the simplex. This latter value is the parameter which is called the

[1]The majority classifier in the Weka system which had been used in our experiments is called ZeroR

number of simplex points on which a simplex is spanned. Thus, the maximum number of the cost function evaluations value has to be increased by the number of simplex points in order to attain the total number of target function calls. For all experiments that have been conducted in our paper we have set the number of simplex points to fifty. The upper limitation on the value of this parameter is the number of samples in the training partition. Therefore, because the smallest problem out of the studied seventeen domains consists of hardly sixty samples, the selected by us value for this parameter seems to be a good choice. The maximum number of cost calls setting of 300 was taken as the default for the datasets of the size of a couple of hundred cases and this choice is based on our earlier experience with similar minimization-based learning systems we had been working on. What concerns the EkP's form of learning used for the Experiment 1, both the first variant of the cost function algorithm involving leave-one-out cross-validation learning as well as the second variant has been employed. The IB1 classifier has been chosen as the EkP's classification engine. Tables 2 and 3 summarize the results of the Experiment 1. It is easy to notice that generalization ability of the EkP system trained with the first algorithm variant depends strongly on the number of prototypes selected. Choosing one prototype per class to be selected by EkP-1 statistically degraded the results with respect to ones obtained with the IB1 system only on three out of the all seventeen domains. This is the excellent result. When two prototypes per class have been selected, the number of times training data reduction degraded the results dropped to only two. With three prototypes per class chosen the results have been statistically insignificant from these attained with IB1 on sixteen problems. The first variant of the EkP algorithm that has been taken for our experiments was trained with leave–one–out cross–validation. The influence of the value of the cross–validation learning fold on the generalization has not been yet fully investigated. Leave–one-out cross–validation seems to lead to obtaining very stable models and the best generalization at the expense of significantly lenghtenning the calculation time. In case of the second algorithm version (EkP-2) statistically significant degradation of the generalization results with respect to ones attained with the IB1 system could have been noted on three datasets independently on the number of prototypes per class chosen.

4.2. Experiment 2: Generalization Ability – LVQ vs. IB1 and LVQ vs. EkP

For this experiment, LVQ version 1 with 'random training data proportional' as well as 'simple k-means' initialization, learning rate of 0.3, total training iterations of 1000, linear decay learning function and disabled voting has been used. Generalization ability of LVQ against IB1 has been tested first. Because the method of initialization of the positions of codebooks seemed not to make any statistically significant influence on generalization of the LVQ system, only one table (Table 4) is provided in which the LVQ system has been used with

Table 2. A comparison of generalization results attained with the E*k*P system with one, two and three prototypes per class selected vs. the generalization obtained with the IB1 classifier. E*k*P has been trained with the first version of the cost function algorithm which is denoted as E*k*P-1. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results with respect to the reference ones (i.e. these of IB1) is marked with a bold font.

#	Dataset	# Classes	IB1	Std. Dev.	EkP-1	Std. Dev.	# P.	EkP-1	Std. Dev.	# P.	EkP-1	Std. Dev.	# P.
1	appendicitis	2	80,28	10,78	86,36	10,25	2	87,18	8,86	4	87,25	8,83	6
2	breast-cancer	2	68,58	7,52	72,98	7,14	2	71,8	6,37	4	72,53	5,97	6
3	horse-colic	2	79,11	6,51	74,62	8,19	2	78,7	5,94	4	78,16	5,87	6
4	credit-rating	2	81,57	4,57	80,2	6,65	2	80,77	5,23	4	81,48	5,36	6
5	german_credit	2	71,88	3,68	69,82	1,9	2	69,59	2,99	4	69,71	3,26	6
6	pima_diabetes	2	70,62	4,67	69,79	5,54	2	70,51	5,37	4	70,4	4,76	6
7	glass	6	69,95	8,43	**57,31**	**9,36**	6	**59,86**	**10,01**	12	62,57	9,51	18
8	cleveland-heart	2	76,06	6,84	80,69	6,54	2	80,72	6,71	4	79,78	6,72	6
9	hungarian-heart	2	78,33	7,54	83,17	6,64	2	82,19	6,79	4	81,64	7,25	6
10	heart-statlog	2	76,15	8,46	81	7,51	2	80,19	7,34	4	80,52	7,34	6
11	hepatitis	2	81,4	8,55	82,29	9,96	2	81,85	9,05	4	82,85	9,13	6
12	labor	2	84,3	16,24	79,93	18,18	2	83,3	16,26	4	84,1	17,3	6
13	lymphography	4	81,54	8,48	74,28	11,43	4	76,55	13,04	8	74,5	10,33	12
14	primary-tumor	21	34,64	7,07	35,69	7,06	21	36,32	8,09	42	35,93	6,87	63
15	sonar	2	86,17	8,45	**66,5**	**9,34**	2	**68,23**	**8,46**	4	**69,47**	**9,86**	6
16	vote	2	92,23	3,95	90,92	3,92	2	92,58	3,93	4	92,43	3,95	6
17	zoo	7	96,55	5,34	**88,72**	**6,77**	7	92,48	6,91	14	94,39	6,75	21
Average		3,76	77,02	7,48	74,96	8,02	3,76	76,05	7,73	7,53	76,34	7,59	11,29
Significance (0.05)						(0/14/3)			(0/15/2)			(0/16/1)	

Table 3. A comparison of the generalization results attained with the E*k*P system with one, two and three prototypes per class selected vs. the generalization obtained with the IB1 classifier. E*k*P has been trained with the second version of the cost function algorithm which is denoted as E*k*P-2. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results with respect to the reference ones (i.e. these of IB1) is marked with a bold font.

#	Dataset	# Classes	IB1	Std. Dev.	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.
1	appendicitis	2	80,28	10,78	85,66	10,6	2	85,62	10,31	4	87,01	9,97	6
2	breast-cancer	2	68,58	7,52	72,98	7,14	2	71,8	6,37	4	72,53	5,97	6
3	horse-colic	2	79,11	6,51	77,03	7,2	2	78,19	6,91	4	78,48	6,14	6
4	credit-rating	2	81,57	4,57	82,09	5,23	2	81,3	5,37	4	81,16	5,01	6
5	german_credit	2	71,88	3,68	69,8	1,9	2	69,93	3,05	4	70,27	2,98	6
6	pima_diabetes	2	70,62	4,67	70,45	6,03	2	70,81	5,79	4	70,71	5,48	6
7	glass	6	69,95	8,43	**57,67**	8,99	6	**60,92**	9,74	12	**61,45**	9,87	18
8	cleveland-heart	2	76,06	6,84	80,47	6,92	2	80,66	6,64	4	79,47	7,29	6
9	hungarian-heart	2	78,33	7,54	82,15	6,68	2	82,35	6,06	4	81,19	6,63	6
10	heart-statlog	2	76,15	8,46	79,63	7,21	2	79,11	6,67	4	79,26	8,17	6
11	hepatitis	2	81,4	8,55	79,79	9,2	2	81,02	9,53	4	82,72	9,57	6
12	labor	2	84,3	16,24	81,07	16,6	2	81,47	16,2	4	82,33	17,23	6
13	lymphography	4	81,54	8,48	75,64	10,77	4	75,32	12,24	8	**74,35**	10,91	12
14	primary-tumor	21	34,64	7,07	35,69	7,06	21	36,32	8,09	42	35,93	6,87	63
15	sonar	2	86,17	8,45	**66,48**	10,15	2	**68,73**	9,47	4	**69,09**	9,75	6
16	vote	2	92,23	3,95	90,92	3,92	2	92,58	3,93	4	92,43	3,95	6
17	zoo	7	96,55	5,34	**88,62**	7,12	7	**92,48**	6,76	14	94,09	6,86	21
Average		3,76	77,02	7,48	75,07	7,81	3,76	75,80	7,83	7,53	76,03	7,80	11,29
Significance (0.05)						(0/14/3)			(0/14/3)			(0/14/3)	

Table 4. A comparison of the generalization results attained with the LVQ-1 system (with the linear decay learning and the training data proportional initialization settings) with 2, 4 and 6 codebooks set vs. the generalization results obtained with the IB1 classifier. The statistical degradation of the results with respect to the reference ones (i.e. these of IB1) is marked by using a bold font.

#	Dataset	# Classes	IB1	Std. Dev.	LVQ (2 P.)	Std. Dev.	LVQ (4 P.)	Std. Dev.	LVQ (6 P.)	Std. Dev.
1	appendicitis	2	80,28	10,78	78,64	15,17	82,72	10,92	85,15	9,37
2	breast-cancer	2	68,58	7,52	66,46	12,18	70,97	4,1	71	4,79
3	horse-colic	2	79,11	6,51	**58,52**	9,88	**62,49**	7,64	**63,75**	7,59
4	credit-rating	2	81,57	4,57	**53,04**	5,39	**58,72**	5,18	**62,35**	5,08
5	german_credit	2	71,88	3,68	66,84	10,94	69,57	4,2	69,78	1,54
6	pima_diabetes	2	70,62	4,67	**61,96**	9,75	66,79	4,32	68,46	5,22
7	glass	6	69,95	8,43	**31,63**	7,96	**33,01**	7,58	**40,15**	9,82
8	cleveland-heart	2	76,06	6,84	**56,52**	8,48	**62,14**	8,94	**62,77**	8,01
9	hungarian-heart	2	78,33	7,54	**62,15**	13	**67,42**	9,39	**65,88**	6,8
10	heart-statlog	2	76,15	8,46	**56,89**	8,23	**62,3**	8,4	**64,41**	8,55
11	hepatitis	2	81,4	8,55	75,39	14,66	78,84	3,88	77,94	4,99
12	labor	2	84,3	16,24	70,6	18,31	84,27	16,88	90,03	12,76
13	lymphography	4	81,54	8,48	**61,16**	15,79	**69,85**	13,31	74,03	10,68
14	primary-tumor	21	34,64	7,07	**12,68**	8,66	**16,02**	8,62	**18,43**	6,88
15	sonar	2	86,17	8,45	**55,34**	8,61	**63,11**	11,67	**67,09**	10,55
16	vote	2	92,23	3,95	**69,71**	20,79	89,84	10,83	93,39	6,71
17	zoo	7	96,55	5,34	**32,99**	12,63	**35,98**	10,33	**36,55**	10,29
Average		3,76	77,02	7,48	57,09	11,79	63,18	8,60	65,36	7,63
Significance (0.05)					(0/5/12)		(0/7/10)		(0/8/9)	

the 'random training data proportional' initialization.

As it can be seen from Table 4, the LVQ system performed rather poorly and on seventeen problems with two codebooks set twelve times statistically significant degradation of the results with respect to these attained with the IB1 classifier has been noted. Increasing the number of codebooks to four has led to a minor improvement of the generalization of the LVQ system and on ten domains the results have been still worse than these obtained with IB1. Selection of six codebooks has led to statistically significant degradation of the results with respect to the reference ones on nine problems out of seventeen studied. In this experiment also no improvement over IB1's generalization ability could have been observed.

In the second experiment in this section the test estimating generalization ability of LVQ against EkP has been performed. This test is made only on two-class problems to assure that the number of LVQ codebooks as well as the prototypes selected by the EkP system is the same. Recall that EkP takes the number of prototypes per class as its adaptive parameter whilst the LVQ system requires a total number of codebooks to be specified. Since all the calculations have been performed in a batch mode with the same settings for all classification domains, the list of datasets had to be restricted to two class problems. What can be noted by taking a closer look at Table 5 is, that the results of LVQ more strongly depend on the number of codebooks selected than it is in case of EkP–1. The average classification accuracy of EkP–1 taken over all twelve domains oscillates around 79% whilst in the case of LVQ, for two codebooks, it equals

Table 5. A comparison of the generalization results attained with the LVQ-1 system with two, four and six codebooks vs. the generalization obtained with the E*k*P classifier. E*k*P has been trained with the first version of the cost function algorithm which is denoted as E*k*P-1. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results of the LVQ system with respect to the reference ones is marked with a bold font.

		2 prototypes (codebooks)				4 prototypes (codebooks)				6 prototypes (codebooks)			
#	Dataset	EkP-1	Std. Dev.	LVQ	Std. Dev.	EkP-1	Std. Dev.	LVQ	Std. Dev.	EkP-1	Std. Dev.	LVQ	Std. Dev.
1	appendicitis	86,36	10,25	78,64	15,17	87,18	8,86	82,72	10,92	87,25	8,83	85,15	9,37
2	breast-cancer	72,98	7,14	66,46	12,18	71,80	6,37	70,97	4,10	72,53	5,97	71,00	4,79
3	horse-colic	74,62	8,19	**58,52**	9,88	78,70	5,94	**62,49**	7,64	78,16	5,87	**63,75**	7,59
4	credit-rating	80,20	6,65	**53,04**	5,39	80,77	5,23	**58,72**	5,18	81,48	5,36	**62,35**	5,08
5	german_credit	69,82	1,90	66,84	10,94	69,59	2,99	69,57	4,20	69,71	3,26	69,78	1,54
6	pima_diabetes	69,79	5,54	**61,96**	9,75	70,51	5,37	66,79	4,32	70,40	4,76	68,46	5,22
7	cleveland-heart	80,69	6,54	**56,52**	8,48	80,72	6,71	**62,14**	8,94	79,78	6,72	**62,77**	8,01
8	hungarian-heart	83,17	6,64	**62,15**	13,00	82,19	6,79	**67,42**	9,39	81,64	7,25	**65,88**	6,80
9	heart-statlog	81,00	7,51	**56,89**	8,23	80,19	7,34	**62,30**	8,40	80,52	7,34	**64,41**	8,55
10	hepatitis	82,29	9,96	75,39	14,66	81,85	9,05	78,84	3,88	82,85	9,13	77,94	4,99
11	labor	79,93	18,18	70,60	18,31	83,30	16,26	84,27	16,88	84,10	17,30	90,03	12,76
12	sonar	66,50	9,34	**55,34**	8,61	68,23	8,46	63,11	11,67	69,47	9,86	67,09	10,55
13	vote	90,92	3,92	**69,71**	20,79	92,58	3,93	89,84	10,83	92,43	3,95	93,39	6,71
Average		78,33	7,83	64,00	11,95	79,05	7,18	70,71	8,18	79,26	7,35	72,46	7,07
Significance (0.05)				(0/5/8)				(0/8/5)				(0/8/5)	

Table 6. A comparison of the generalization results attained with the LVQ-1 system with two, four and six codebooks vs. the generalization obtained with the E*k*P classifier. E*k*P has been trained with the second version of the cost function algorithm which is denoted as E*k*P-2. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results of the LVQ system with respect to the reference ones is marked with a bold font.

		2 prototypes (codebooks)				4 prototypes (codebooks)				6 prototypes (codebooks)			
#	Dataset	EkP-2	Std. Dev.	LVQ	Std. Dev.	EkP-2	Std. Dev.	LVQ	Std. Dev.	EkP-2	Std. Dev.	LVQ	Std. Dev.
1	appendicitis	85,66	10,60	78,64	15,17	85,62	10,31	82,72	10,92	87,01	9,97	85,15	9,37
2	breast-cancer	72,98	7,14	66,46	12,18	71,80	6,37	70,97	4,10	72,53	5,97	71,00	4,79
3	horse-colic	77,03	7,20	**58,52**	9,88	78,19	6,91	**62,49**	7,64	78,48	6,14	**63,75**	7,59
4	credit-rating	82,09	5,23	**53,04**	5,39	81,30	5,37	**58,72**	5,18	81,16	5,01	**62,35**	5,08
5	german_credit	69,80	1,90	66,84	10,94	69,93	3,05	69,57	4,20	70,27	2,98	69,78	1,54
6	pima_diabetes	70,45	6,03	**61,96**	9,75	70,81	5,79	66,79	4,32	70,71	5,48	68,46	5,22
7	cleveland-heart	80,47	6,92	**56,52**	8,48	80,66	6,64	**62,14**	8,94	79,47	7,29	**62,77**	8,01
8	hungarian-heart	82,15	6,68	**62,15**	13,00	82,35	6,06	**67,42**	9,39	81,19	6,63	**65,88**	6,80
9	heart-statlog	79,63	7,21	**56,89**	8,23	79,11	6,67	**62,30**	8,40	79,26	8,17	**64,41**	8,55
10	hepatitis	79,79	9,20	75,39	14,66	81,02	9,53	78,84	3,88	82,72	9,57	77,94	4,99
11	labor	81,07	16,60	70,60	18,31	81,47	16,20	84,27	16,88	82,33	17,23	90,03	12,76
12	sonar	66,48	10,15	**55,34**	8,61	68,73	9,47	63,11	11,67	69,09	9,75	67,09	10,55
13	vote	90,92	3,92	**69,71**	20,79	92,58	3,93	89,84	10,83	92,43	3,95	93,39	6,71
Average		78,35	7,60	64,00	11,95	78,74	7,41	70,71	8,18	78,97	7,55	72,46	7,07
Significance (0.05)				(0/7/6)				(0/8/5)				(0/8/5)	

only 64%. Going with the number of codebooks to four and six, increases the average LVQ's generalization ability to about 70% and 72% respectively. Similar trends can be observed when LVQ is put against the EkP–2 (see Table 6).

4.3. Experiment 3: Time Requirements

The training times of the EkP system, which are however all statistically worse than these of IB1 (it is not a surprise), are quite short and in average are equal to about 1s (EkP–1) and 0.2s (EkP-2) for learning on a single partition of a typical UCI dataset of a size of a couple of hundred cases (see Table 7 and 8).[2] The training times of LVQ are even shorter than these obtained with our system. As can be seen from Table 9, LVQ has beaten up completely both variants of the EkP method on all seventeen classification problems. It turned out that the LVQ system can be trained in time which is of three orders of magnitude shorter than the one obtained by measuring the EkP's learning time. Fortunately the EkP testing times are shorter than these of IB1 by three orders of magnitude. Table 10 contains the summary of the results of the measurements of the testing time. It is not hard to see that it takes much less than a minute for the entire 10-fold cross-validation test that is conducted with our system to complete on most common UCI datasets. This is acceptable result. It should be noted that training the EkP method with lower-fold cross-validation than leave–one–out leads to a significant reduction of the time requirements for this algorithm.

5. Conclusions

We are lucky that we have managed to create quite a fast prototype selection system despite of employing the simplex minimization routine which is usually expensive. The initial experiments indicate that the method may turn out to be competitive to other data pruning systems. In the preliminary calculations the method discussed in this paper have shown statistical insignificance of the generalization ability with respect to IB1 almost on all classification problems and sometimes turned out to be superior to the LVQ system ver. 1. However the EkP training times are longer that these of IB1 and of LVQ but the testing times are shorter than the ones obtained by timing IB1. After all, one should remember about the general idea laying behind the selection of prototypes: once the instances are initially found (training sets are pruned), the tests on unseen samples which are usually frequently performed can be conducted much faster. Before the EkP system is not confronted with many other prototype selection algorithms and before further experiments with our method are not performed it will be hard to estimate a real value of our contribution to the pattern recognition field.

[2] The calculations have been performed on a laptop equipped with a 2.4GHz Intel Core 2 Duo processor running 64–bit Ubuntu Linux Operating System under 64–bit OpenJVM Java 1.6.

Table 7. The training times of the E*k*P method attained on one cross–validation fold in seconds. E*k*P has been trained with the first version of the cost function algorithm which is denoted as E*k*P-1. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results of the E*k*P system with two and three prototypes per class selected with respect to the reference ones (i.e. these of E*k*P–1 with one reference instance per class chosen) is marked with a bold font.

#	Dataset	EkP-1	Std. Dev.	# P.	EkP-1	Std. Dev.	# P.	EkP-1	Std. Dev.	# P.
1	appendicitis	0,094100	0,011024	2	**0,122640**	0,012439	4	**0,150680**	0,015624	6
2	breast-cancer	0,387710	0,031026	2	**0,463360**	0,031208	4	**0,539880**	0,029821	6
3	horse-colic	0,896440	0,038249	2	**1,288820**	0,051652	4	**1,671100**	0,054170	6
4	credit-rating	2,157560	0,652252	2	2,416530	0,078107	4	**2,841530**	0,072034	6
5	german_credit	4,069850	0,102912	2	**5,012590**	0,130176	4	**5,901870**	0,114146	6
6	pima_diabetes	2,101360	0,069882	2	**2,326480**	0,061336	4	**2,559110**	0,074534	6
7	glass	0,451500	0,024611	6	**0,701010**	0,036583	12	**0,947280**	0,040618	18
8	cleveland-heart	0,743410	0,040017	2	**1,140030**	0,046655	4	**1,681980**	0,335402	6
9	hungarian-heart	0,702100	0,039398	2	**1,062410**	0,044892	4	**1,432560**	0,061142	6
10	heart-statlog	0,445890	0,031254	2	**0,589730**	0,031850	4	**0,736540**	0,039326	6
11	hepatitis	0,260750	0,025523	2	**0,395390**	0,036889	4	**0,525430**	0,037316	6
12	labor	0,074720	0,014501	2	**0,114500**	0,015386	4	**0,153150**	0,015815	6
13	lymphography	0,345840	0,022860	4	**0,576120**	0,036620	8	**0,802680**	0,040138	12
14	primary-tumor	2,895140	0,083248	21	**5,342370**	0,135357	42	**7,802860**	0,140862	63
15	sonar	1,465180	0,061098	2	**2,772280**	0,332526	4	**3,837320**	0,091913	6
16	vote	0,922810	0,040775	2	**1,191440**	0,045587	4	**1,456910**	0,052481	6
17	zoo	0,309320	0,030384	7	**0,543520**	0,031581	14	**0,779980**	0,041131	21
Average		1,077864	0,077589	3,76	1,532895	0,068167	7,53	1,989462	0,073910	11,29
Significance (0.05)						(0/1/16)			(0/0/17)	

Table 8. The training times of the E*k*P method attained on one cross–validation fold in seconds. E*k*P has been trained with the second version of the cost function algorithm which is denoted as E*k*P-2. Fifty simplex points have been used to train the E*k*P system. The statistical degradation of the results of the E*k*P system with two and three prototypes per class selected with respect to the reference ones (i.e. these of E*k*P–2 with one reference instance per class chosen) is marked with a bold font.

#	Dataset	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.
1	appendicitis	0,037790	0,007664	2	**0,045880**	0,007532	4	**0,055930**	0,011434	6
2	breast-cancer	0,082680	0,010414	2	**0,107170**	0,016144	4	**0,126710**	0,013139	6
3	horse-colic	0,163910	0,014278	2	**0,243600**	0,021487	4	**0,310910**	0,031048	6
4	credit-rating	0,251590	0,027058	2	**0,349290**	0,021823	4	**0,445370**	0,041481	6
5	german_credit	0,399330	0,022377	2	**0,587730**	0,033918	4	**0,741370**	0,035401	6
6	pima_diabetes	0,227630	0,021423	2	**0,299020**	0,026785	4	**0,347850**	0,019753	6
7	glass	0,125850	0,013469	6	**0,197960**	0,016508	12	**0,264570**	0,027457	18
8	cleveland-heart	0,165890	0,014700	2	**0,261310**	0,018811	4	**0,344170**	0,020931	6
9	hungarian-heart	0,152880	0,014194	2	**0,241970**	0,029088	4	**0,315260**	0,030305	6
10	heart-statlog	0,103060	0,010329	2	**0,141760**	0,014758	4	**0,171600**	0,014837	6
11	hepatitis	0,072560	0,017038	2	**0,099140**	0,010503	4	**0,124040**	0,012784	6
12	labor	0,030960	0,007271	2	**0,040720**	0,007770	4	**0,051060**	0,011704	6
13	lymphography	0,087670	0,010470	4	**0,135960**	0,013544	8	**0,180570**	0,015458	12
14	primary-tumor	0,562650	0,033226	21	**1,036940**	0,046220	42	**1,466880**	0,061435	63
15	sonar	0,245350	0,018604	2	**0,377860**	0,034716	4	**0,488430**	0,024053	6
16	vote	0,150160	0,014686	2	**0,202640**	0,016651	4	**0,250320**	0,026978	6
17	zoo	0,083240	0,010366	7	**0,133620**	0,012904	14	**0,179630**	0,016602	21
Average		0,173129	0,015739	3,76	0,264857	0,020539	7,53	0,344981	0,024400	11,29
Significance (0.05)						(0/0/17)			(0/0/17)	

Table 9. The training times of the EkP method attained on one cross–validation fold in seconds. EkP has been trained with the first and the second version of the cost function algorithm which is denoted as EkP–1 and EkP-2 respectively. Two codebooks /prototypes have been chosen. Fifty simplex points have been used to train the EkP system. The statistical degradation of the results of the EkP system with respect to the reference ones (i.e. these of LVQ) is marked with a bold font.

#	Dataset	LVQ (2 c.)	Std. Dev.	EkP-1 (2 p.)	Std. Dev.	EkP-2 (2 p.)	Std. Dev.
1	appendicitis	0,001200	0,005662	**0,094100**	0,011024	**0,037790**	0,007664
2	breast-cancer	0,000890	0,000399	**0,387710**	0,031026	**0,082680**	0,010414
3	horse-colic	0,001720	0,000570	**0,896440**	0,038249	**0,163910**	0,014278
4	credit-rating	0,001810	0,000526	**2,157560**	0,652252	**0,251590**	0,027058
5	german_credit	0,002260	0,000543	**4,069850**	0,102912	**0,399330**	0,022377
6	pima_diabetes	0,000980	0,000426	**2,101360**	0,069882	**0,227630**	0,021423
7	glass	0,000820	0,000479	**0,451500**	0,024611	**0,125850**	0,013469
8	cleveland-heart	0,001260	0,000441	**0,743410**	0,040017	**0,165890**	0,014700
9	hungarian-heart	0,001260	0,000463	**0,702100**	0,039398	**0,152880**	0,014194
10	heart-statlog	0,001050	0,000261	**0,445890**	0,031254	**0,103060**	0,010329
11	hepatitis	0,001270	0,000446	**0,260750**	0,025523	**0,072560**	0,017038
12	labor	0,001090	0,000321	**0,074720**	0,014501	**0,030960**	0,007271
13	lymphography	0,001310	0,000545	**0,345840**	0,022860	**0,087670**	0,010470
14	primary-tumor	0,001370	0,000506	**2,895140**	0,083248	**0,562650**	0,033226
15	sonar	0,003450	0,000575	**1,465180**	0,061098	**0,245350**	0,018604
16	vote	0,001270	0,000468	**0,922810**	0,040775	**0,150160**	0,014686
17	zoo	0,001980	0,007790	**0,309320**	0,030384	**0,083240**	0,010366
Average		0,001470	0,001201	1,077864	0,077589	0,173129	0,015739
Significance (0.05)				(0/0/17)		(0/0/17)	

Table 10. The testing times of the EkP method attained on one cross–validation test fold in seconds. EkP has been trained with the second version of the cost function algorithm which is denoted as EkP-2. Fifty simplex points have been used to train the EkP system. The statistical **improvement** of the results of the EkP system with respect to the reference ones (i.e. these of IB1) is marked with a bold, italic font.

#	Dataset	IB1	Std. Dev.	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.	EkP-2	Std. Dev.	# P.
1	appendicitis	0,000350	0,000479	***0,000000***	0,000000	2	0,000040	0,000197	4	0,000010	0,000100	6
2	breast-cancer	0,002740	0,000441	***0,000070***	0,000256	2	***0,000070***	0,000256	4	***0,000090***	0,000288	6
3	horse-colic	0,011370	0,000485	***0,000130***	0,000338	2	***0,000140***	0,000349	4	***0,000280***	0,000451	6
4	credit-rating	0,030640	0,009157	***0,000110***	0,000314	2	***0,000280***	0,000451	4	***0,000410***	0,000494	6
5	german_credit	0,081050	0,010622	***0,000410***	0,000494	2	***0,000590***	0,000494	4	***0,000690***	0,000465	6
6	pima_diabetes	0,022770	0,001874	***0,000140***	0,000349	2	***0,000260***	0,000441	4	***0,000270***	0,000446	6
7	glass	0,002010	0,000301	***0,000080***	0,000273	6	***0,000130***	0,000338	12	***0,000220***	0,000416	18
8	cleveland-heart	0,006040	0,008393	0,000230	0,000423	2	0,000250	0,000435	4	0,000290	0,000456	6
9	hungarian-heart	0,004350	0,000479	***0,000070***	0,000256	2	***0,000150***	0,000359	4	***0,000250***	0,000435	6
10	heart-statlog	0,004320	0,000490	***0,000090***	0,000288	2	***0,000110***	0,000314	4	***0,000130***	0,000338	6
11	hepatitis	0,001950	0,000261	***0,000100***	0,000302	2	***0,000040***	0,000197	4	***0,000140***	0,000349	6
12	labor	0,000200	0,000402	0,000010	0,000100	2	0,000050	0,000219	4	0,000020	0,000141	6
13	lymphography	0,001700	0,000503	***0,000060***	0,000239	4	***0,000100***	0,000302	8	***0,000100***	0,000302	12
14	primary-tumor	0,006670	0,000533	***0,000580***	0,000496	21	***0,001030***	0,000171	42	***0,001680***	0,000649	63
15	sonar	0,012170	0,000877	***0,000120***	0,000327	2	***0,000410***	0,000494	4	***0,000460***	0,000501	6
16	vote	0,012060	0,009183	***0,000150***	0,000359	2	***0,000170***	0,000378	4	***0,000200***	0,000402	6
17	zoo	0,000670	0,000473	***0,000100***	0,000302	7	***0,000060***	0,000239	14	***0,000140***	0,000349	21
Average		0,011827	0,002644	0,000144	0,000301	3,76	0,000228	0,000331	7,53	0,000316	0,000387	11,29
Significance (0.05)				(15/2/0)			(14/3/0)			(14/3/0)		

References

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

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

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

Brownlee J.: A java implementation of the SOM-LVQ PAK. http://www.it.swin.edu.au/personal/jbrownlee/

Cameron-Jones R.: Instance selection by encoding length heuristic with random mutation hill climbing. In: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence, (1995), 99-106

Dobosz K.: Statistical Significance Tests in Estimation of the Results Obtained with Various Systems that Learn. M.Sc. thesis, Nicholaus Copernicus University, Toruń, Poland, (2006) (In Polish)

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

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

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

Grochowski M.: Selecting Reference Vectors in Selected Methods for Classification. M.Sc. thesis, Nicholaus Copernicus University, Department of Applied Informatics, Toruń, Poland, (2003) (In Polish)

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

Grudzinski K., Duch W.: SBL-PM: A Simple Algorithm for Selection of Reference Instances for Similarity-Based Methods. Intelligent Information Systems, Bystra, Poland, 2000, in Advances in Soft Computing, Physica-Verlag, (2000), 99-108

Grudzinski K.: SBL-PM-M: A System for Partial Memory Learning. Artificial Intelligence and Soft Computing ICAISC 2004, in Lecture Notes in Artificial Intelligence (LNAI **3070**), 586-591

Grudzinski K.: E*k*P: A fast minimization based prototype selection algorithm. Proceedings of the International IIS'08 Conference, Zakopane, Poland, 2008. In: Challenging Problems of Science, Computer Science. Academic Publishing House EXIT, Warsaw, (2008), 45-53

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

Hyninen, Kangas, Kohonen, Laaksonnen, Torkolla.: LVQ_PAK: The Learning Vector Quantization Program Package, (1996)

Jankowski N.: Data regularization. In Rutkowski, L., Tadeusiewicz, R., eds.: Neural Networks and Soft Computing, Zakopane, Poland, (2000), 209-214

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

Kohonen T.: Self-Organizing Maps. Third ed. Berlin Heidelberg. Springer-Verlag, (2001). (Thomas S Huang; Teuvo Kohonen, and Manfred R. Schroeder. Springer Series in Information Sciences, **30**).

Lampton M.: neldermead.java (http://www.cea.berkeley.edu/ mlampton/neldermead.java)

Maloof M., Michalski R.: Selecting Examples for Partial Memory Learning. Machine Learning, **41**, (2000), 27-52

Mertz C., Murphy P.: UCI repository of machine learning databases. http://www.ics.uci.edu/pub/machine-learning-data-bases.

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

Oja M., Kaski S., Kohonen T.: Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001 Addendum, Neural Computing Surveys, **3**, (2003), 1-156

Skalak D.: Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: International Conference on Machine Learning (1994), 293-301

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

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

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

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

Witten I., Frank E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, (2000).