Control and Cybernetics

vol. **39** (2010) No. 2

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 new system for selection of reference instances, which is called the EkP system (**E**xactly k **P**rototypes), has been introduced by us recently. In this paper we study suitability of the EkP method for training data reduction on seventeen datasets. As the underlaying 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, Learning Vector Quantization, for which the same number of codebooks has been chosen as the number of prototypes selected by the EkP system. The comparison indicates that even with only a few prototypes which have been chosen by the EkP method on nearly all seventeen datasets statistically indistinguishable results from those given by the IB1 system have been obtained. On many datasets generalization ability of the EkP method has been higher than the one attained with LVQ.

Keywords: classification of data, selection of reference vectors, prototype methods.

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 or sometimes as prototypes, can be either selected from training data or can be generated out of a training set. In the latter case the instance features have in general different values from the ones stored in the original training set. The idea behind these both two techniques is that only a small set obtained from a usually much larger,

[*]Submitted: January 2009; Accepted: May 2009.

original training set, is used for a final classification of unseen samples (Maloof and Michalski, 2000; Martinez and Wilson, 1997, 2000; Grochowski, 2003; Jankowski and Grochowski, 2004; Grochowski and Jankowski, 2004; Duch and Grudzinski, 2000; Grudzinski, 2004, 2008).

Prototype selection is an important problem which has been frequently studied in machine learning and pattern recognition. Selection of reference instances can significantly speed up classification and analysis of data 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 it is often the other way around, i.e. data pruning improves the generalization ability of classifiers.

This paper concerns the first of above mentioned problems, i.e. 'instance selection', 'training data compression, reduction or pruning'. For this purpose a new system, which we call EkP, has been introduced recently (Grudzinski, 2008). The acronym EkP stands for **E**xactly-**k**-**P**rototypes. Our method is based on minimization of the number of errors, which the employed underlying classifier makes during learning on a training partition. In other words, building the EkP model (i.e. training it) on training data involves performing a process of minimization. Thus, in order to conduct a classic 10-fold cross-validation test, minimization has to be repeated 10 times: one time for every training partition. This minimization process results in construction of the optimal new target training set containing k prototype instances per each class occurring in a problem domain. The parameter k, which controls the number of samples selected is given by a user of the program. The aim of the cost function, which is called from a minimization procedure and whose pseudocode is given in detail further in the paper, is to extract k instances per class from the vector of optimization parameters (they are selected randomly for the parameter vector from the original training set) and to form from them a new small training set whose content is optimized with respect to the generalization ability of the employed classifier. Thus, the purpose of the cost function is to decode the vector of optimization parameters, extract the instances, build from these samples a new small learning set, train on it the underlying classifier and return the number of classification errors made during learning. After completion of minimization, the underlying classifier is built on a training set consisting of just the found prototypes and after that its generalization ability is estimated on unseen samples.

We want to stress here that our new system differs significantly from our earlier model, PM-M (Grudzinski, 2004), but it is common to both systems that they are minimization based. Samples selected with the EkP system can be used, for example, to build prototype-based rules, introduced in Duch and Grudzinski (2001) and Blachnik and Duch (2004), and which are an alternative to the classic logical rules.

2. Methodologies for reference instance 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 Grochowski (2003).

2.1. Formulation of the problem

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. This technique should result in minimal degradation of the generalization ability of the underlying classifier or, ideally, ought to lead to the improvement of the classification results with respect to these that are obtained by using the original set $\mathcal{T}$ for learning. In other words, reference case selection is a method for selection or generation of the most informative samples (called by us also supervectors) 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. Let us denote by n the number of samples of the original training set $\mathcal{T}$. 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 the optimal subset $\mathcal{S}$ out of all possible $2^n - 1$ subsets with respect to generalization ability of the underlaying classifier.

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 outliers 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, 1972), All k-NN (Tomek, 1976) and ENRBF (Jankowski, 2000) are the key examples of the algorithms in 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 sacrifice of generalization of the employed underlying classifiers. Condensation methods aim at seeking and removing 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 lay far from the decision borders. CNN (Hart, 1968), RNN (Gates, 1972), GA, RNGE (Bhattacharya, Poulsen and Toussaint, 1981), ICF (Brighton and Mellish, 2002) and DROP 1-5 (Martinez and Wilson, 2000) are the main systems that fall 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 samples carrying particularly large amount of information and capable of representing large number of cases are known as prototype methods. In case when $\mathcal{S}$ is a subset of $\mathcal{T}$ these methods are called prototype selection algorithms, otherwise prototype generation systems. The difference between data condensation algorithms and prototype selection methods is very subtle. In our understanding, prototype selection 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. Both groups of methods, though, try to arrive at the smallest set $\mathcal{S}$, the stress in data condensation techniques is put on generalization, whilst in the case of prototype selection 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 number of samples selected, may be treated either as data condensation methods or as prototype selection models. LVQ (Kaski, Kohonen and Oja, 2003), MC1 and RMHC (Skalak, 1994), IB3 (Aha, Albert and Kibler, 1991), ELH, ELGrow and Expolore (Cameron-Jones, 1995) and our own models PM-M (Grudzinski, 2004) and E*k*P (Grudzinski, 2008) can be included into the prototype selection and generation group of methods.

3. The E*k*P system

The E*k*P system is based on minimization of a cost function, which returns the number of errors the classifier makes. Despite this, the E*k*P method is relatively fast because during every evaluation of the cost function the reduced training set is constructed only out of the preset small number of instances. It takes seconds for the E*k*P method to perform 10-fold cross-validation on most common UCI datasets. In our implementation we used the well known simplex method (Nelder and Mead, 1965) for function minimization, which we have taken from the Internet (Lampton, 2004).

The simplex must be initialized first before a minimization procedure is started. The E*k*P system is very sensitive to the way in which the simplex is initialized, and therefore we have decided to include in the text of the paper the description of the E*k*P's initialization algorithm, which is given below.

Algorithm 1 The EkP's simplex initialization algorithm

```
Require: A vector of training set instances trainInstances[] (numInstances
  dimensional)
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
    randomize(trainInstances[])
    for $j$ = 0 to numInstances - 1 do
      for $k$ = 0 to numAttributes - 1 do
        simplex[j][k] = trainInstances[j][k]
      end for
    end for
    $c = 0$
    for $j$ = 0 to numClasses * numProtoPerClass - 1 do
      for $k$ = 0 to numAttributes - 1 do
        $p$[k + numAttributes * j] = simplex[c][k]
      end for
      if $c$ ¡= numInstances then
        $c = c + 1$
      else
        $c = 0$
      end if
    end for
    simplex[$i$][numAttributes] = costFunction($p$[])
  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 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 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 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

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

```
Require: A vector of training set instances 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 at 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 associated with it the 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 vector of training set instances 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 at 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 associated with it the lowest value of
  numClassificationErrors
  return numClassificationErrors
```

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 *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 the 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 and Murphy, 1996) have been performed. The information about the datasets used can be found in Table 1.

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

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 (1-Nearest Neighbor) (Aha, Albert and Kibler, 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, Albert and Kibler, 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 calculations with extremely low number of prototypes selected may lead to good generalization of the method on unseen samples. The well known LVQ method (Hyninen et al., 1996; Kohonen, 2001; Kaski, Kohonen and Oja, 2003), even though it is 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.

Ten-fold stratified cross-validation test has been performed for all seventeen domains. In the experiments conducted with the EkP system in each cross-validation fold the training partition has been pruned so that only the prototype cases remained. Then the EkP's underlying classifier has been trained and its 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, taken over all the available hundred partial results, have been reported.

Generalization ability of the EkP 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 IB1 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.

The single corrected re-sampled T-Test (Witten and Frank, 2000; Dobosz, 2006) has been used to calculate statistical significance of the results (with the factor of 0.05) in order to help making decision whether the EkP 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, 2004). Finally, what remains to be mentioned is that the EkP system has been written by the author in Java as the contribution to the SBLWeka project (Grudzinski, 2005).

4.1. Experiment 1: generalization ability – EkP 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 EkP, are not provided here. 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 *appendicitis*, *breast-cancer*, *german-credit* and *hepatitis* datasets the results have been statistically insignificant.

The EkP 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

[1]The majority classifier of the Weka system, which was used in our experiments is called ZeroR.

to 300 calls, excluding a certain number of evaluations required to initialize the simplex. This latter value is the parameter, which is called the 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 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 value for this parameter, selected by us, seems to be a good choice. The maximum number of cost calls 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 form of learning used for Experiment 1, both the first variant of the cost function algorithm involving leave-one-out cross-validation learning as well as the second variant have been employed. The IB1 classifier has been chosen as the EkP classification engine.

Tables 2 and 3 summarize the results of 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. Selection of one prototype per class by the EkP-1 system statistically degraded the classification results with respect to ones obtained with the IB1 system only on three out of the all seventeen domains. 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, see Table 2. 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 fully investigated yet. Leave-one-out cross-validation seems to lead to very stable models and the best generalization at the expense of significantly lengthening the calculation time. In case of the second variant of the algorithm (EkP-2) statistically significant degradation of the generalization results with respect to ones attained with the IB1 system could be noted on three datasets independently of the number of prototypes per class chosen, see Table 3.

Table 2. Comparison of generalization results attained with the EkP system with one, two and three prototypes per class selected vs. the results from the IB1 classifier. EkP was trained with the first version of the cost function algorithm, denoted EkP-1. Fifty simplex points were used to train the EkP system. Statistical degradation of results with respect to the reference (i.e. 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	71.80	6.37	4	72.53	5.97	6
3	horse-colic	2	79.11	6.51	74.62	8.19	2	78.70	5.94	4	78.16	5.87	6
4	credit-rating	2	81.57	4.57	80.20	6.65	2	80.77	5.23	4	81.48	5.36	6
5	german_credit	2	71.88	3.68	69.82	1.90	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	70.40	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	81.00	7.51	2	80.19	7.34	4	80.52	7.34	6
11	hepatitis	2	81.40	8.55	82.29	9.96	2	81.85	9.05	4	82.85	9.13	6
12	labor	2	84.30	16.24	79.93	18.18	2	83.30	16.26	4	84.10	17.30	6
13	lymphography	4	81.54	8.48	74.28	11.43	4	76.55	13.04	8	74.50	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	**66.50**	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. Comparison of the generalization results attained with the EkP system with one, two and three prototypes per class selected vs. the results from the IB1 classifier. EkP was trained with the second version of the cost function algorithm, denoted EkP-2. Fifty simplex points were used to train the EkP system. Statistical degradation of results with respect to the reference (i.e. 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	10.60	2	85.62	10.31	4	87.01	9.97	6
2	breast-cancer	2	68.58	7.52	72.98	7.14	2	71.80	6.37	4	72.53	5.97	6
3	horse-colic	2	79.11	6.51	77.03	7.20	2	78.19	6.91	4	78.48	6.14	6
4	credit-rating	2	81.57	4.57	82.09	5.23	2	81.30	5.37	4	81.16	5.01	6
5	german_credit	2	71.88	3.68	69.80	1.90	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	81.40	8.55	79.79	9.20	2	81.02	9.53	4	82.72	9.57	6
12	labor	2	84.30	16.24	81.07	16.60	2	81.47	16.20	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)		

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 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 a statistically significant degradation of results with respect to those 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 those obtained with IB1. Selection of six codebooks has led to statistically significant degradation of 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 has been observed.

Table 4. Comparison of the generalization results attained with the LVQ-1 system (with the linear decay learning and the training data proportional initialization settings) with two, four and six codebooks set vs. the results from the IB1 classifier. Statistical degradation of results with respect to the reference (i.e. IB1) is marked with 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	4.10	71.00	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	4.20	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**	13.00	**67.42**	9.39	**65.88**	6.80
10	heart-statlog	2	76.15	8.46	**56.89**	8.23	**62.30**	8.40	**64.41**	8.55
11	hepatitis	2	81.40	8.55	75.39	14.66	78.84	3.88	77.94	4.99
12	labor	2	84.30	16.24	70.60	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)	

In the second experiment in this section the test estimating generalization ability of LVQ against EkP has been performed. This test was 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 depend more strongly on the number of codebooks selected than it is in case of E*k*P-1. The average classification accuracy of E*k*P-1, taken over all twelve domains oscillates around 79% whilst in the case of LVQ, for two codebooks, it equals only 64%. Increasing 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 E*k*P-2 (see Table 6).

Table 5. Comparison of the generalization results attained with the LVQ-1 system with two, four and six codebooks vs. the results from the E*k*P classifier. E*k*P was trained with the first version of the cost function algorithm, denoted E*k*P-1. Fifty simplex points were used to train the E*k*P system. Statistical degradation of results of the LVQ system with respect to the reference 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. Comparison of the generalization results attained with the LVQ-1 system with two, four and six codebooks vs. the results from the E*k*P classifier. E*k*P was trained with the second version of the cost function algorithm, denoted E*k*P-2. Fifty simplex points were used to train the E*k*P system. Statistical degradation of results of the LVQ system with respect to the reference 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	67.09
13	vote	90.92	3.92	**69.71**	20.79	92.58	3.93	89.84	10.83	92.43	3.95	93.39	93.39
Average		78.33	7.60	64.00	11.95	78.74	7.41	70.71	8.18	78.97	7.55	72.46	18.09
Significance (0.05)				(0/7/6)				(0/8/5)				(0/8/5)	

4.3. Experiment 3: time requirements

The training times of the EkP system, although all statistically worse than those 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 Tables 7 and 8)[2]. The training times of LVQ are even shorter than those obtained with our system. As it can be seen from Table 9, LVQ has outpaced 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 EkP. Fortunately, the EkP testing times are shorter than those of IB1 by two orders of magnitude.

Table 10 contains the summary of results of the measurements of 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 an 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 have developed a relatively fast prototype selection system despite 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 has shown statistically insignificant difference of the generalization ability with respect to IB1 on almost all classification problems and sometimes turned out to be superior to the LVQ system ver. 1. However, the EkP training times are longer that those of IB1 and of LVQ, though the testing times are shorter than the ones obtained by timing IB1. After all, one should remember about the general idea lying 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. The results of the EkP system are promising, but further experiments with our method and comparison with other prototype selection methods must be performed in order to have better knowledge about the value of the proposed method in 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. Training times (in seconds) of the EkP method attained on one cross-validation fold. EkP was trained with the first version of the cost function algorithm, denoted EkP-1. Fifty simplex points were used to train the EkP system. Statistical degradation of the results of the EkP system with two and three prototypes per class selected with respect to the reference (i.e. EkP-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. Training times (in seconds) of the EkP method attained on one cross-validation fold. EkP was trained with the second version of the cost function algorithm, denoted EkP-2. Fifty simplex points were used to train the EkP system. Statistical degradation of the results of the EkP system with two and three prototypes per class selected with respect to the reference (i.e. EkP-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		0.273129	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. Training times (in seconds) of the E*k*P method attained on one cross-validation fold. E*k*P was trained with the first and second versions of the cost function algorithm, denoted E*k*P-1 and E*k*P-2, respectively. Two codebooks /prototypes have been chosen. Fifty simplex points were used to train the E*k*P system. Statistical degradation of the results of the E*k*P system with respect to the reference (i.e. LVQ) is marked with a bold font.

#	Dataset	LVQ (2c.)	Std. Dev.	EkP-1 (2 p.)	Std. Dev.	EkP-2 (2p.)	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. Testing times (in seconds) of the E*k*P method attained on one cross-validation test fold. E*k*P was trained with the second version of the cost function algorithm, denoted E*k*P-2. Fifty simplex points were used to train the E*k*P system. The statistical **improvement** of the results of the E*k*P system with respect to the reference (i.e. IB1) is marked with a bold, italic font.

#	Dataset	LB1	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	0.002470	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**	0.000107	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	3.7	0.000228	0.000331	7.5	0.000316	0.000387	11.2
						6			3			9
Significance (0.05)				(15/2/0)			(14/3/0)			(14/3/0)		

References

AHA, D., KIBLER, D. and ALBERT, M. (1991) Instance-based learning algorithms. *Machine Learning* **6**, 37-66.

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

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

BROWNLEE, J. (2004) A java implementation of the SOM-LVQ PAK. http://www.it.swin.edu.au/personal/jbrownlee/, http://wekaclassalgos.sourceforge.net

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

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

DUCH, W. and BLACHNIK, M. (2004) Fuzzy rule-based systems derived from similarity to prototypes. **LNCS 3316**, 912-917.

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

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

GROCHOWSKI, M. (2003) Selection of Reference Vectors in Selected Methods for Classification. M.Sc. thesis, Nicolaus Copernicus University, Department of Applied Informatics, Toruń, Poland, (in Polish).

GROCHOWSKI, M. and JANKOWSKI, N. (2004) Comparison of Instance Selection Algorithms II: Results and Comments. *Artificial Intelligence and Soft Computing ICAISC 2004*, **LNAI 3070**, Springer, 580-585.

GRUDZINSKI, K. and DUCH, W. (2000) 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, 99-108.

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

GRUDZINSKI, K. (2005) SBLWeka: A modified Weka System. http://scientific-activity-karol-grudzinski.blogspot.com.

GRUDZINSKI, K. (2008) EkP: A fast minimization based prototype selection algorithm. *Proceedings of the International IIS'08 Conference*, Zakopane, Poland, 2008. Academic Publishing House EXIT, Warsaw, 45-53.

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

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

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

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

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

Lampton, M. (no date) neldermead.java . The version of the original neldermead.java code modified by the author of this paper that has been used for the calculations used in this paper can be found at http://scientific-activity-karol-grudzinski.blogspot.com.

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

Mertz, C. and Murphy, P. (1996) UCI repository of machine learning databases. http://archive.ics.uci.edu/ml/.

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

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

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

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

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: *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.

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