Combining the Principal Components Method with Different Learning Algorithms

Luboš Popelínský

Faculty of Informatics, Masaryk University
Brno, Czech Republic
popel@fi.muni.cz

Abstract. Principal components (PCO's) analysis is an unsupervised projection method that computes linear combinations of the original attributes that are constructed to explain maximally the variance. We show that the method is useful as a pre-processing step for various learning algorithms. Adding PCO's as new attributes results in decrease of error rate for a significant number of data sets for C5.0 decision tree learner, instance-based learner, and naive Bayes classifiers. It the case of C5.0 the decrease of error rate can be achieved without an increase of decision tree complexity. Replacement of attributes with PCO's results in increase of accuracy only for naive Bayes classifier.

1 Introduction

In the process of learning it is very often transformation and reduction of data, that play an important role. A good transformation/reduction technique may result in new attributes that are more appropriate for a learning algorithm. A wide range of constructive induction techniques [3, 4, 16] has been explored aiming at enriching the language of propositional learners.

Principal components (PCO's) [9, 13] are useful linear functions that capture linear dependencies among attributes. This data preprocessing technique serves frequently for better understanding (e.g. visualisation) or compressing the data [5, 6]. To our best knowledge few systematic studies have been carried out to evaluate their benefits and the associated computational costs. The aim of this work is to contribute to this issue.

We have shown in [10, 11] that if knowing the query examples (attribute values of a test set) combination of principal components method and decision tree learner C5.0 leads to a decrease in error rate without an increase in hypothesis complexity (the size of a decision tree). Here we bring results for the usual setting when query examples are not known in advance. We show that also for this settings some decrease of error rate was observed for several learning algorithms.

The method presented here consists of two steps. In the first step we compute the principal components from numerical attributes. In the second step we use these principal components to add them to the original attributes. The first step is actually a form of constructive induction [3] applicable for numerical variables, which uses the methodology of the principal component analysis to generate derived attributes.

We show that for three learning algorithms – C5.0, an instance-based learner and naive Bayes – addition of PCO's displays decrease of error rate for a significant number of data sets. For C5.0 we introduce new criteria that say when PCO's have such an effect and what profit can be expected.

The paper is organised as follows. In Section 2 we briefly overview main ideas of principal components. Information about data set can be found in Section 3. The method used is described in Section 4. General overview of experimental results can be found in Section 5. Discussion of results follows in Section 6.

2 Principal Components Method

Principal components analysis [9, 13] (or the Karhunen-Loeve expansion) is a statistical method for dimensionality reduction. Principal components are commonly used to *replace* attributes with smaller number of new attributes. It is an unsupervised projection method that computes linear combinations of the original attributes that are constructed to explain maximally the variance. Let $X = (X_1, ..., X_n)$ be a random vector with variance matrix V. When looking for the first principal component c_1 we look for a vector $c_1 = (c_{11}, ..., c_{1n})$ such that $c_1 c_1^T = 1$ and variance of $c_1 X^T$ is maximal. It means that $c_1 X^T$ describes as much of variablity of data X as possible. The next principal component is computed in a similar way, and must be uncorrelated with the first one, etc. Thus the first few principal components are supposed to contain most of the information implicit in the attributes. It can be shown [9, 13] that the principal components c_i are equal to eigenvectors of the variance matrix V.

3 Data sets

The datasets used represent a subset of datasets used in comparative studies within project METAL [2] (October 2000). We explored all data sets that do not contain missing values. We also removed from our experiments the data sets for which most learners reached accuracy better than 99% as no profit can be obtained for such data. Because of dimensionality problems (too many attributes or too many records ≥ 20000) `musk` and `pyrimidines` data set were removed, too. We evaluated both methods on 17 datasets containing both numerical and categorical attributes.

4 Method

Learners. We explored propositional learners that are frequently used in machine learning community: decision tree learner C5.0 without boosting – `c50tree`, instance-based learner `mlcib` [1] and naive Bayes classifier `mlcnb` [1, 8]. For all learners the default settings of parameters has been used.

Algorithm. All results below are computed using 10-fold cross validation. In each step the following algorithm was called:

1. Normalise data (i.e. continuous attributes);
2. Compute the first N principal components (linear functions) from the learning set;
3. Employ these linear functions to compute PCO's for the learning set as new attributes;
   ADD the computed values of principal components to the learning set;
4. Normalise the test set using means and standard deviations that have been computed from the learning set;
5. Employ the same linear functions as above to compute PCO's for the test set;
   ADD the computed values of the principal components to the test set;
6. Run a learning algorithm on the extended learning set ;
7. Test the result on the extended test set.

We tested all possible values of $N = 1..Number\_of\_Attributes$ for each data set.

5 Results

In Fig. 8 there are results for four data sets and three learners `c50tree, mlcib, mlcnb` that display typical trends of increase/decrease of accuracy. Four typical trends can be observed that are independent on a particular learner. For some data sets PCO's displayed no effect – when increasing their number error rate was not changing (`diabetes` for all three learners; `letter` for `c50tree`). Sometimes the trend of error rate was decreasing but it never decrease below the default level – error rate on the data set without PCO's (`letter` and `c50tree`). The most interesting trends can be seen for `satimage` and `waveform21` data sets. For small number of PCO's – 1 or 2 – the error rate is higher but for 3,4,... PCO's it decrease below the default level (`satimage` for any of the three learners; `waveform21` for `c50tree, mlcnb`). The important fact is that this trend is typical for majority of data sets for which an extension of data with PCO's results in increase of accuracy.

Tables 1, 2, 3 contain results for C5.0 without boosting (`c50tree`), instance-based learner `mlcib`, and naive Bayes classifier `mlcnb` when PCO's have been added to the original data. In the first three columns there are results for the optimal number of PCO's – the number of them, comparison with error rate on the original data (++ – decrease of error rate $> 95\%$, + – decrease of error rate $\leq 95\%$), error rate ratio (error rate for the extended data set divided by the error rate for the original one), and error rate for the extended data. The second

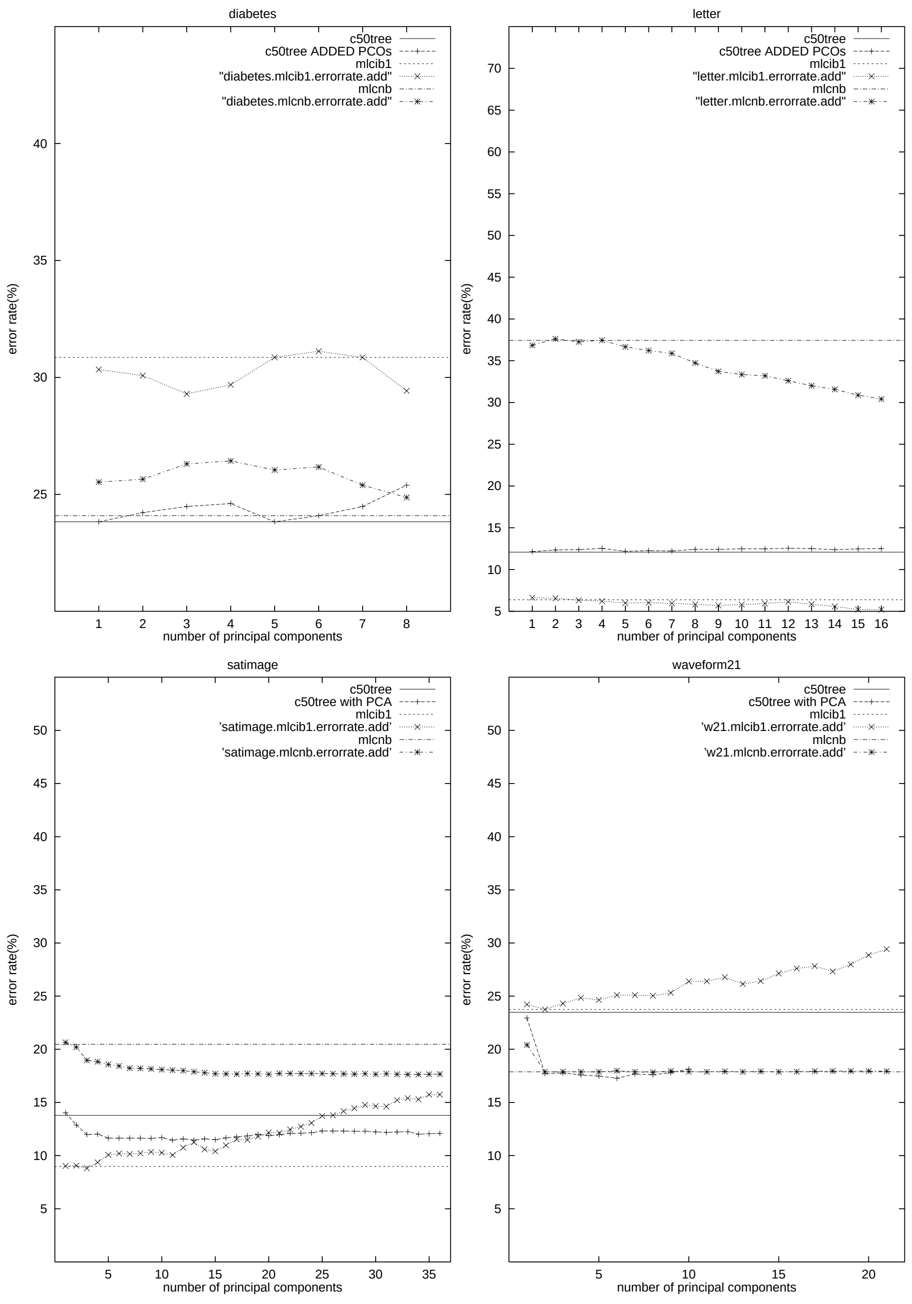


Fig. 1. Results for `diabetes, letter, satimage, waveform21`

Table 1. Results for C5.0 without boosting (`c50tree`)

						c50tree+					c50tree
	best N of PCO's				added 2 PCO's			added 3 PCO's			
	N		e.r.ratio	err.rate		e.r.ratio	err.rate		e.r.ratio	err.rate	
ann	1		1.06	0.35		1.303	0.43		1.303	0.43	0.33
australian	1		1.000	15.51		1.009	15.65		1.000	15.51	15.51
diabetes	1		1.00	23.83		1.016	24.21		1.016	24.48	23.83
fluid*	2	++	0.94	2.79	++	0.94	2.79	++	0.94	2.79	2.98
german_num*	3	+	0.980	28.80	+	0.986	29.00	+	0.980	28.80	29.40
letter	1		1.01	12.16		1.021	12.34		1.024	12.38	12.09
optical	1	++	0.91	8.36	++	0.921	8.47	++	0.925	8.51	9.20
page	3	+	0.99	3.05		1.010	3.12	+	0.99	3.05	3.09
pendigits	12	++	0.893	3.24	+	0.986	3.58	++	0.915	3.32	3.63
quisclas	6	+	0.984	37.14		1.014	38.28	+	0.998	37.67	37.75
satimage	11	++	0.838	11.45	++	0.943	12.88	++	0.877	11.98	13.66
segment	3	+	0.964	3.51		1.022	3.72	+	0.964	3.51	3.64
vehicle	2		1.009	26.48		1.009	26.48		1.027	26.95	26.24
vowel	4	++	0.837	18.18	++	0.889	19.29	++	0.870	18.89	21.71
waveform21	5	++	0.728	17.10	++	0.732	17.20	++	0.729	17.12	23.48
waveform40	6	++	0.689	17.28	++	0.707	17.72	++	0.709	17.78	25.08
yeast	2		1.014	43.87		1.014	43.87		1.025	44.34	43.26

Table 2. Results for `mlcib` instance-based learner

						mlcib+					mlcib
	best N of PCO's				added 2 PCO's			added 3 PCO's			
	N		e.r.ratio	err.rate		e.r.ratio	err.rate		e.r.ratio	err.rate	
ann	4	+	0.99	2.32		1.02	2.37		1.01	2.36	2.33
australian	2	+	0.994	22.03	+	0.994	22.03		1.020	22.61	22.17
diabetes	3	++	0.949	29.30	+	0.975	30.08	++	0.949	29.30	30.86
fluid*	5	++	0.821	4.28	++	0.923	4.84	++	0.893	4.65	5.21
german_num*	3	+	0.968	30.80	+	0.974	31.00	+	0.968	30.80	31.8
letter	16	++	0.815	5.20		1.028	6.56	+	0.992	6.33	6.38
optical	9	++	0.886	3.02	+	0.977	3.33	+	0.956	3.26	3.41
page	2	+	0.971	3.73	+	0.971	3.73	+	0.990	3.80	3.84
pendigits	1	+	0.956	0.66		1.000	0.691		1.000	0.69	0.69
quisclas	3	+	0.99	41.08	+	0.99	41.27	+	0.99	41.08	41.54
satimage	3	+	0.981	8.81		1.006	9.03	+	0.981	8.81	8.98
segment	11	+	0.951	5.63		1.103	6.53		1.074	6.36	5.92
vehicle	16	++	0.797	24.11		1.00	30.38		1.05	31.68	30.26
vowel	6	++	0.54	0.71	++	0.84	1.11	++	0.92	1.21	1.31
waveform21	2	+	0.97	23.74	+	0.97	23.74	+	0.99	24.30	24.58
waveform40	3	+	0.95	28.36	+	0.96	28.61	+	0.95	28.36	29.84
yeast	4	+	0.95	47.03	+	0.98	48.38	+	0.98	48.45	49.33

and the third triple of columns differ only in the first column. The last column of the tables displays error rates for the original data set.

First we have to answer the question about usefulness of PCO's for particular learner. For `c50tree` (Tab. 1), for 11 out of 17 data sets decrease of error rate has been observed. It means that a decrease of error rate appeared for significant number of data sets (one-sample Wilcoxon test on the level 95%). However, the decrease is not significant on level $\geq$ 90% (t-test). It should be noticed that none of accuracy but `waveform21` is smaller than the accuracy for C5.0 with boosting.

For an instance-based learner `mlcib1` we can see (first triple of columns in Tab 2) that for any data set the error rate with PCO's is smaller than error rate for original data Using t-test, hypothesis that PCO's do not influence error rate can be rejected on level 90% (T=2.042, n=16). We can conclude that extension of a data set with PCO's results in a significantly lower error rate. Unfortunately, a similar assertion does not hold for 3 PCO's.

As can be seen from Tab. 3, even for naive Bayes the method results if decrease of error rates for most of data sets. It is significant on level 99% (one-sample Wilcoxon test) but the decrease of error rate is not significant on level $\geq$ 90% (t-test).

In the next section we mainly discuss results obtained with C5.0.

Table 3. Results for `mlcnb` naive Bayes

	mlcnb+										mlcib
	best N of PCO's				added 2 PCO's			added 3 PCO's			
	N		e.r.ratio	err.rate		e.r.ratio	err.rate		e.r.ratio	err.rate	
ann	6	++	0.90	4.06	++	0.905	4.08	++	0.94	4.24	4.51
australian	1		1.077	24.35		1.090	24.64		1.078	24.35	22.61
diabetes	8		1.032	24.87		1.065	25.65		1.092	26.30	24.09
fluid*	9	++	0.919	8.56		1.000	9.31	++	1.000	9.31	9.31
german_num*	1		1.004	24.40		1.004	24.40		1.062	25.80	24.30
letter	16	++	0.854	30.42		1.056	37.62		1.045	37.25	35.63
optical	10	+	0.971	9.38		1.006	9.72	+	0.997	9.63	9.66
page	5	++	0.727	7.29	++	0.919	9.22	++	0.821	8.24	10.03
pendigits	10	++	0.791	11.69		1.032	14.73	++	0.947	13.51	14.27
quisclas	16	+	0.971	58.83	+	0.997	60.43		0.998	60.48	60.60
satimage	34	++	0.861	17.64	+	0.986	20.20	++	0.926	18.96	20.48
segment	11	++	0.840	16.88		1.036	20.82		1.012	20.34	20.09
vehicle	17	++	0.71	37.71		1.02	54.14		1.01	53.55	53.07
vowel	10	++	0.84	27.37			33.03			34.74	32.52
waveform21	2	++	0.90	17.20	++	0.90	17.20	++	0.94	17.90	19.08
waveform40	9	++	0.93	18.50	++	0.93	18.50	++	0.93	18.56	19.96
yeast	7	+	0.96	45.55			46.36			45.96	47.57

6 Discussion

For what data set are PCO's useful ? The statement above can be reformulated by such a way: for most of data sets there exists a number N of PCO's that are good candidates for adding to the original set of attributes. A question that needs to be answered now is: what is the optimal number N of PCO's that we should add? One way of finding N is based on eigenvalues of variance matrix. In [10, 11] N was set the number of PCO's to the number of eigenvalues that are significantly greater than one. It can be computed using the following criterion[14] :

$$N \;=\; \#eigenvalues|eigenvalue > 1 + 2 * \sqrt{\frac{\#attributes-1}{\#examples-1}}$$

Another possibility is to exploit the level of variance that is explained with the first N principal components. Let $X = (X_1, ..., X_n)$ be again a random vector with variance matrix V. Let $Z_i,\ i = 1, ..., r$ be principal components, r is a number of positive eigenvalues of variance matrix V. Then the following equalities hold:

$$varZ_1 + varZ_2 + ... + varZ_r = varX_1 + varX_2 + ... + varX_n$$

As $varZ_i = \lambda_i$, where λ_i is the i-th eigenvalue of V, the previous equation can be rewritten as:

$$\lambda_1 + \lambda_2 + ... + \lambda_r = varX_1 + varX_2 + ... + varX_n$$

The sum $\sigma^2 = varX_1 + varX_2 + ... + varX_n$ can be seen as a variability measure of vector X. Thus to explain X in terms of Z we need such number N of principal components that $(\lambda_1 + ... + \lambda_N)/\sigma^2$ is close to 1.

It was already mentioned above that for most of data sets a trend of accuracy as a function of N has a similar shape. For small number of PCO's – 1 or 2 – the error rate is higher than the error rate of `c50tree` but starting from 3 PCO's the error rate decreases below this level. Compare now the first and the third triple of columns in Tab. 1. We can see that all data sets that are promising for any value of N are also promising for $N = 3$. It does not mean at all that for some $N > 3$ the error rate ratio will be even smaller. However, the expected difference will not be big (see Tab 1). To prevent from over-fitting we prefer to add as little of attributes as possible. Thus $N = 3$ seems be a good choice.

Let us validate this hypothesis. As showed above $(\lambda_1 + ... + \lambda_N)/\sigma^2$ displays a quality of approximation via first N PCO's. For the 15 of 17 data sets explored it was observed that if a sum of the first three PCO's express more than three fifth of variability

$$(\lambda_1 + \lambda_2 + \lambda_3)/\sigma^2 \;>\; 0.6$$

then increase of accuracy appears. It is significant on level 99% (Sign test).

In the following text we call a *promising data set* (for a given learner) if the error rate on the extended data is smaller than for the original set of attributes.

How a hypothesis size changed? We compared the sizes of decision trees (number of nodes) with the corresponding sizes of trees when using PCO's. For 13 out of 17 datasets the size of the decision tree generated decreased when using principal components. From promising data sets only for `fluid` data the tree complexity overcame the complexity of tree without PCO's. Some small increase in hypothesis complexity was observed for `fluid` from 15.4 to 16.3 (5.2%) what corresponds to 1 extra node in the decision tree. More extra nodes appeared for `australian` (increase from 19.9 to 21.8, Δ=9.5%, i.e. approx. 2 extra nodes), `yeast` (166.3, 181.5, Δ=9.1%, 15 nodes) and `diabetes` (22.1, 25.4, Δ=14.9%, approx. 3 extra nodes). When adding the complexity of 3 PCO's (number of attributes * 3) to the complexity of the tree this sum was smaller than `c50tree` complexity only for `waveform21`.

Where PCO's appear in a decision tree? We observed that at least one of PCO's appeared in the root (on level 1) of the decision tree or not deeper than on level 4 for the promising data sets. It holds for 15 out of 17 data sets (`australian` was removed because no PCO's appeared in the tree) (significant for the Sign test on level 99%) It must be noticed that it is the 2nd principal components that appears "sooner" : for 7 out of 11 such data sets it appeared 7-times (1st PCO three-times, 3rd PCO twice) as the first in the decision tree).

Table 4. Training time

	c50tree+		c50tree
	PCO time	train. time	time
ann	0.71	0.56	0.43
australian	0.04	0.07	0.04
fluid	0.18	0.11	0.15
diabetes	0.06	0.08	0.05
letter		10.05	7.68
optical	0.85	4.36	7.42
pendigits	3.96	1.02	3.37
page	0.52	0.82	1.14
quisclas	1.87	2.44	2.53
satimage	4.22	3.73	4.55
segment	0.75	0.57	0.61
vehicle	0.31	0.23	0.24
vowel	0.34	0.74	0.51
waveform21	0.20	1.87	2.55
waveform40	1.79	3.68	5.46
yeast	0.23	0.43	0.60

Training time. In our analysis of times (Tab. 4), we first compared the training with and without PCO's. For 12 out of 17 data sets the training time for data with PCO's was smaller than for the same data without PCO's. From promising data sets, only `vowel` displayed greater training time for data with PCO's. Then the time to compute PCO's was added to the training time of the decision tree and compared with the time used to construct a decision tree without PCO's. Only for `ann` data set the ratio of these times overcame 3, and only for 5 out of 16 the ratio was greater 2, i.e. more than twice slower than without PCO's. For `optical` data the sum of PCO time and the training time was smaller than the time without PCO's. Thus, we can conclude that time complexity seems acceptable.

Other learners We also performed first experiments with other learners, namely with C5.0 with boosting, with linear discriminant method, and with linear trees (a decision tree that can introduce linear function in its nodes) [7]. However, no significant decrease of erro rate has been observed for them. The reason may be that they actually form some kind of linear combination of attributes.

7 Conclusion

We described a method that employed the principal components analysis as a pre-processing stage for propositional learners. To summarise our results:

- adding principal components to the original dataset results in a decrease in error rate for significant number of datasets for C5.0, instance-based learner and naive Bayes learner;
- a decrease of error rate is significant for the instance-based learner;
- for c5.0 decrease of error rate can be achieved without increasing complexity of the decision tree;

The naive Bayes is the only algorithm where replacement of attributes with PCO's leads to decrease of error rate. It happened for 11 out of 17 data sets.

Some of results are pretty preliminary, e.g. criterion for usefulness of PCO's, and need to be evaluated on other data sets. Neither we compared PCO's with other preprocessing methods for attribute reduction.

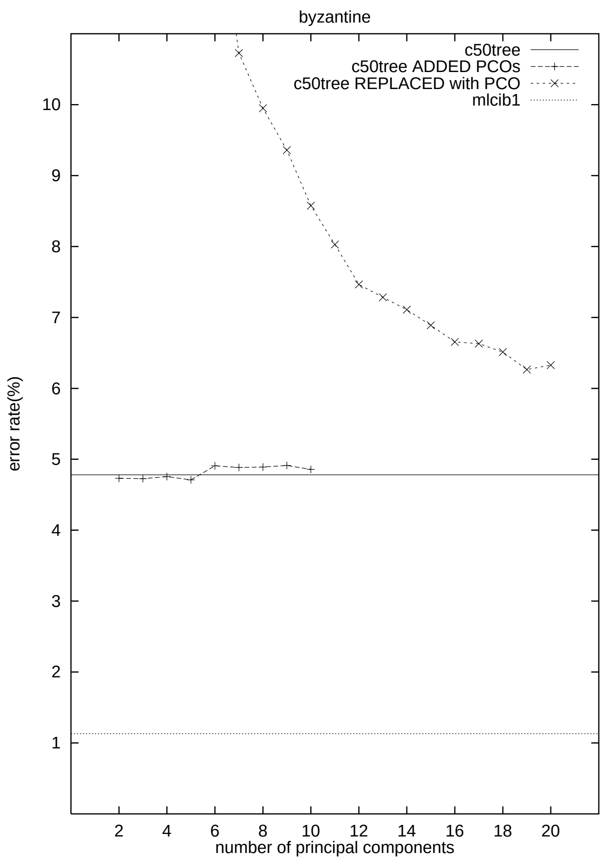
Further work should be carried out to explore some other way of exploitation of principal components, e.g. pruning them (simplifying the linear functions) or choosing only some of them. No theoretical analysis has been performed yet to answer these questions. Exploitation of other statistical methods, e.g. correspondence analysis, is also a challenge.

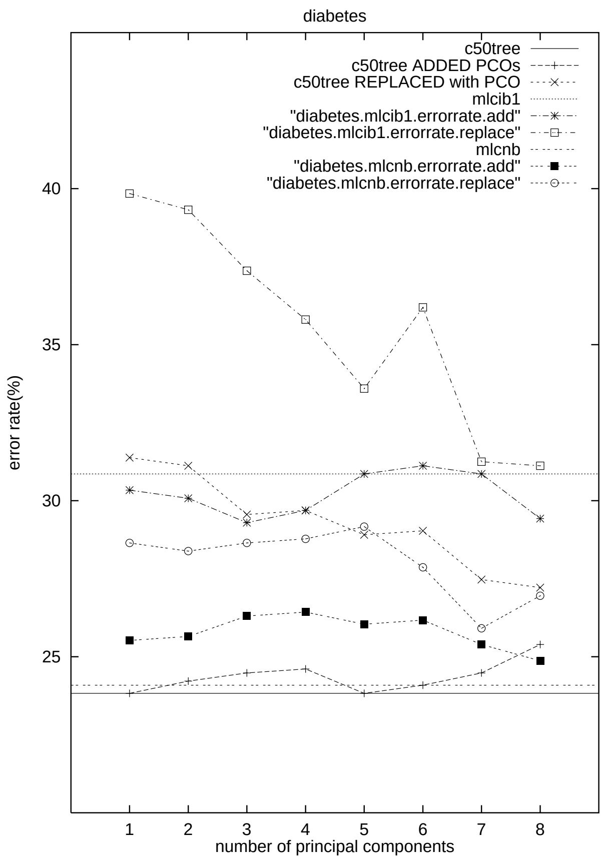
Acknowledgements

Thanks to Pavel Brazdil and other members of LIACC of University of Porto for their help. This work has been partially supported by ESPRIT METAL Project.

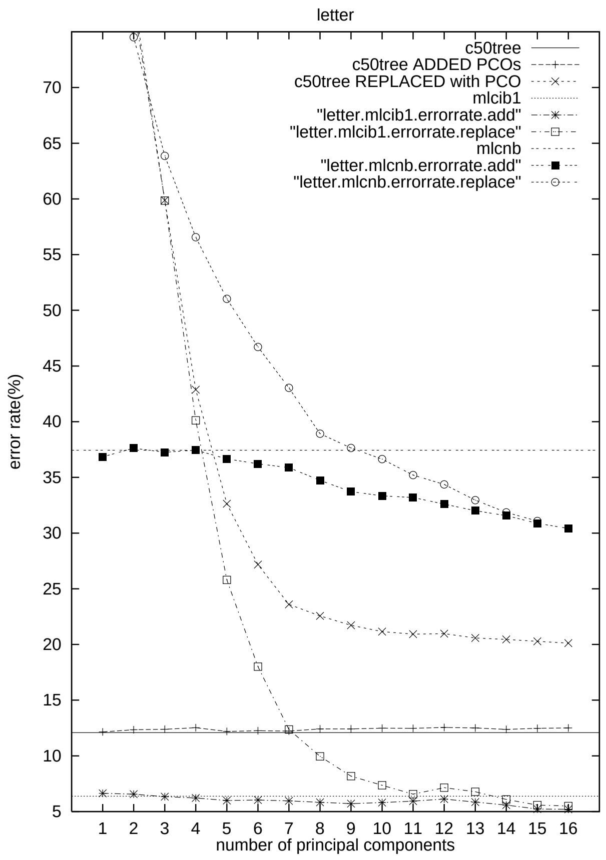
References

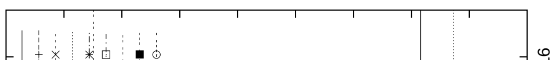
1. Machine Learning Library in C++ `http://www.sgi.com/Technology/mlc/`
2. One of pages of Esprit LTR Project METAL `http://www.ncc.up.pt/liacc/ML/METAL/`
3. Bala J.W., Michalski R.S., Wnek J.: The Principal Axes Method for Constructive Induction. *In Sleeman D. and Edwards P.(eds.), Machine Learning: Proceedings of the Ninth International Workshop (ML92), Morgan Kaufmann, Los Altos/Palo Alto/San Francisco*, pp.20-29, 1992.
4. Bloedorn, E. and Michalski, R.S.: Data-Driven Constructive Induction. *IEEE Intelligent Systems, Special issue on Feature Transformation and Subset Selection*, pp. 30-37, March/April, 1998.
5. Z. Duszak and W.W. Koczkodaj, Using Principal Component Transformation in Machine Learning, *Proceedings of International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden Germany*, pp. 125-129, 1994.
6. A. Famili, Wei-Min Shen, Richard Weber, Evangelos Simoudis: Data Preprocessing and Intelligent Data Analysis. *Intelligent Data Analysis Vol 1., No.1, 1997.*
7. Gama J.: Probabilistic linear tree. *In D. Fisher (ed.), Proceedings of the 14th Int. Conf. on Machine Learning (ICML'97)*, Morgan Kaufman, 1997.
8. Mitchell, T.M.: *Machine Learning.* McGraw Hill, New York, 1997.
9. F. Murtagh and A. Heck: *Multivariate Data Analysis.* Kluwer Academic, Dordrecht, 1987.
10. Popelínský L., Brazdil P.: Combining the Principal Components Method with Decision Tree Learning. In *Proceedings of Multistrategy Learning Workhop MSL-2000*, Guimarães, Portugal 2000.
11. Popelínský L., Brazdil P.: The Principal Components Method as a Pre-processing Stage for Decision Tree Learning. *PKDD'2000 Ws on Metalearning*, Lyon 2000
12. Quinlan J.R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publ. 1992.
13. Ripley B.D.: *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.
14. Saporta G.: Some simple rules for interpreting outputs of principal components and correspondence analysis. *In: Bacelar-Nicolau H., Nicolau F.C., Janssen J.(eds.): Proceedings of 9th Intl.Symp. on Applied Stochastic Models and Data Analysis ASMDA-99, University of Lisbon*, 1999.
15. Torgo L.: Inductive Learning of Tree-based Regression Models. PhD Thesis, Faculty of Science University of Porto, 1999.
16. Wnek, J. and Michalski, R.S.: Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments. *Machine Learning, Vol. 14, No. 2*, pp. 139-168, 1994.

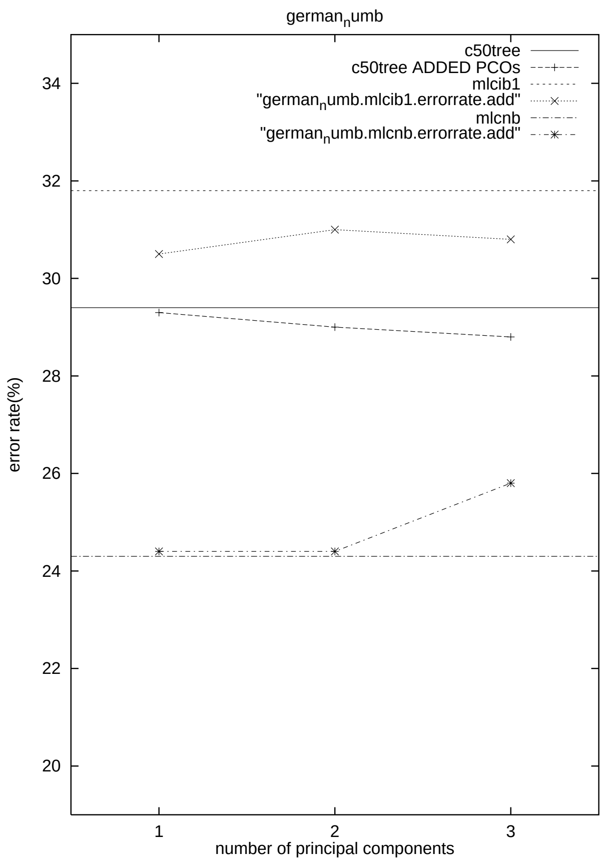

byzantine
c50tree
c50tree ADDED PCOs
c50tree REPLACED with PCO
mlcib1
error rate(%)
number of principal components
1
2
3
4
5
6
7
8
9
10
2
4
6
8
10
12
14
16
18
20

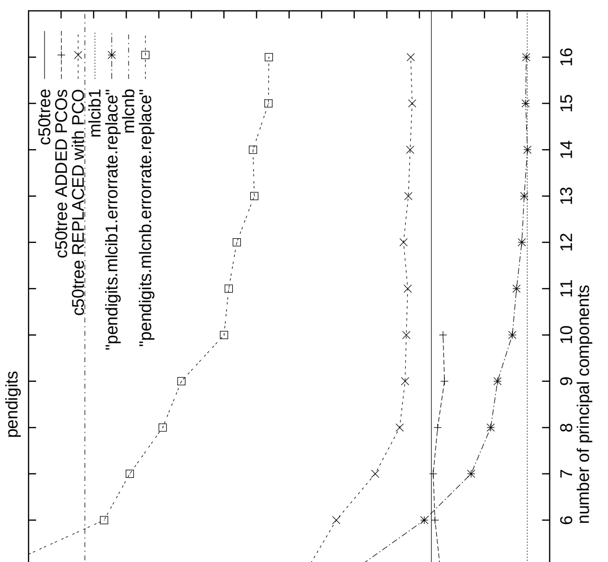


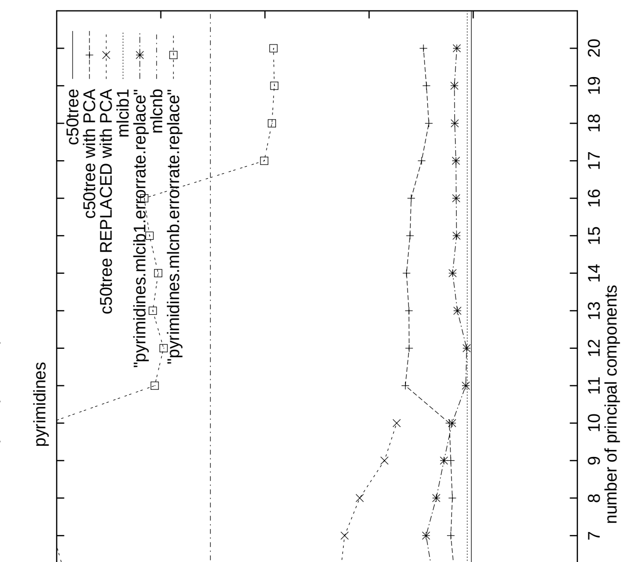
8 Appendix A: Results

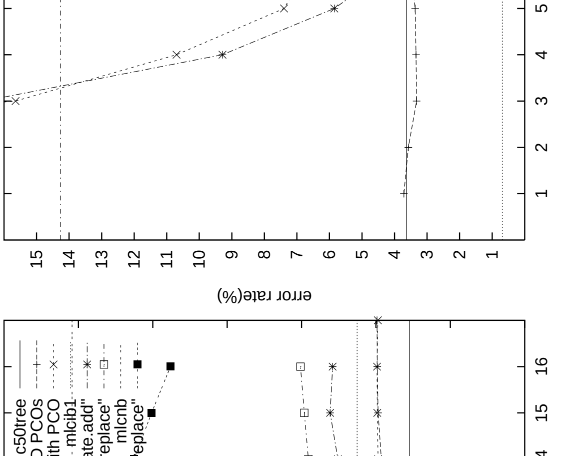


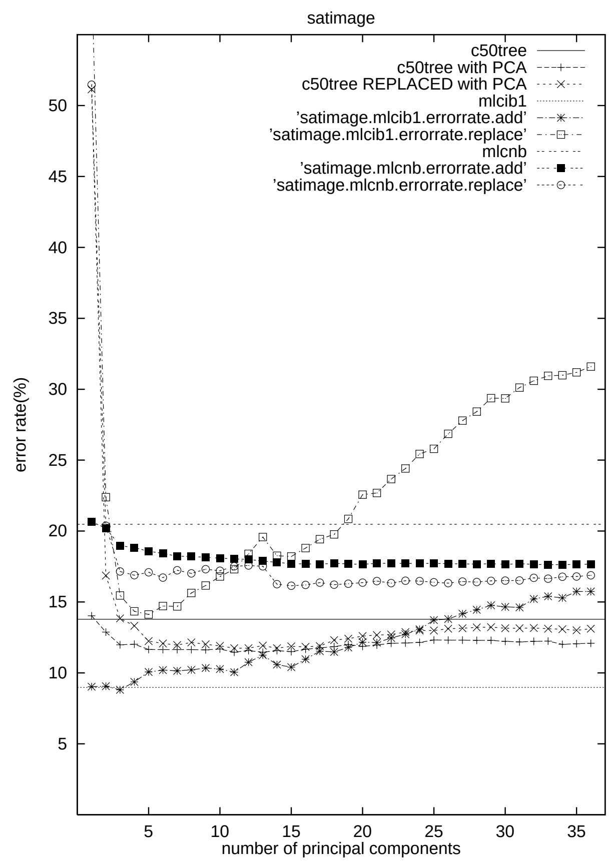


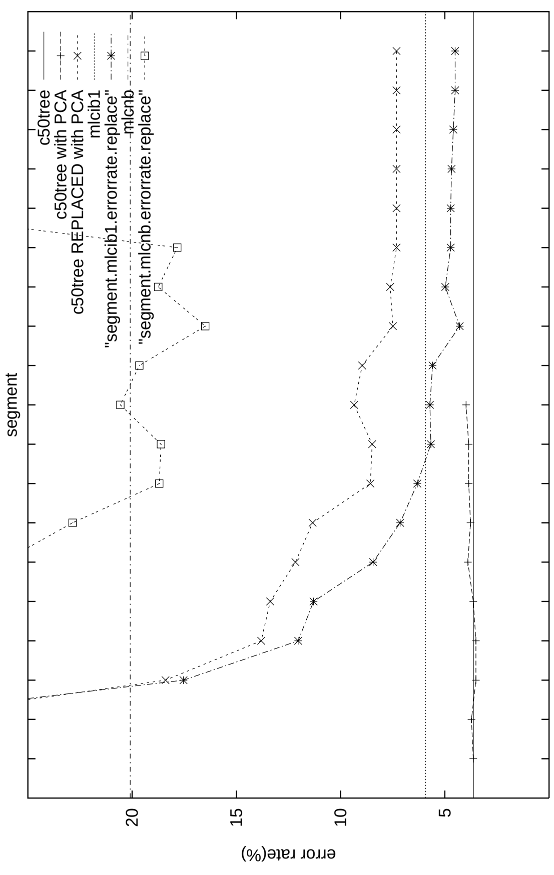

german_numb
c50tree
c50tree ADDED PCOs
mlcib1
"german_numb.mlcib1.errorrate.add"
mlcnb
"german_numb.mlcnb.errorrate.add"
error rate(%)
number of principal components

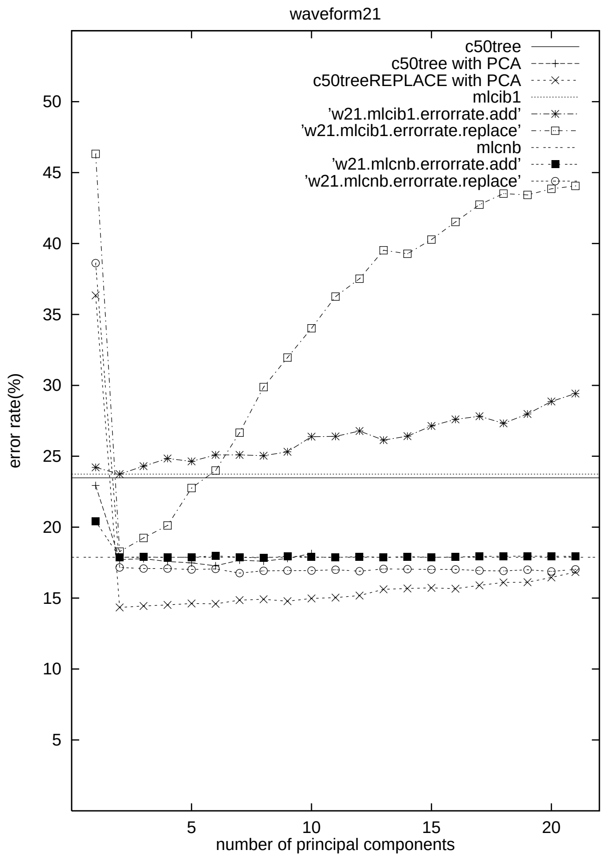

pendigits
c50tree
c50tree ADDED PCOs
c50tree REPLACED with PCO
mlcib1
"pendigits.mlcib1.errorrate.replace"
mlcnb
"pendigits.mlcnb.errorrate.replace"
error rate(%)
number of principal components

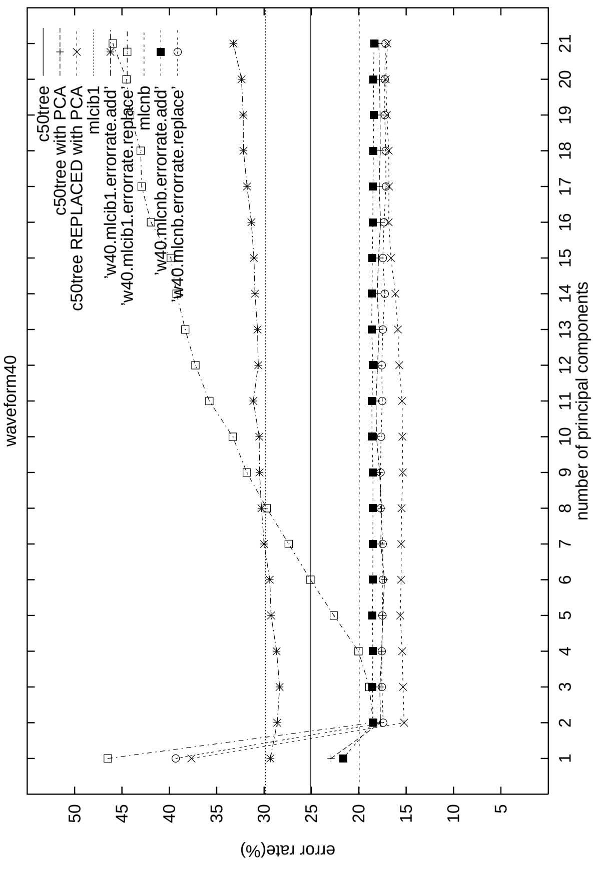

pyrimidines
c50tree
c50tree with PCA
c50tree REPLACED with PCA
mlcib1
"pyrimidines.mlcib1.errorrate.replace"
mlcnb
"pyrimidines.mlcnb.errorrate.replace"
error rate(%)
number of principal components


c50tree


satimage
c50tree
c50tree with PCA
c50tree REPLACED with PCA
mlcib1
'satimage.mlcib1.errorrate.add'
'satimage.mlcib1.errorrate.replace'
mlcnb
'satimage.mlcnb.errorrate.add'
'satimage.mlcnb.errorrate.replace'
error rate(%)
number of principal components


segment
c50tree
c50tree with PCA
c50tree REPLACED with PCA
mlcib1
"segment.mlcib1.errorrate.replace"
mlcnb
"segment.mlcnb.errorrate.replace"
error rate(%)


waveform21
c50tree
c50tree with PCA
c50treeREPLACE with PCA
mlcib1
'w21.mlcib1.errorrate.add'
'w21.mlcib1.errorrate.replace'
mlcnb
'w21.mlcnb.errorrate.add'
'w21.mlcnb.errorrate.replace'
error rate(%)
number of principal components


waveform40
c50tree
c50tree with PCA
c50tree REPLACED with PCA
mlcib1
'w40.mlcib1.errorrate.add'
'w40.mlcib1.errorrate.replace'
mlcnb
'w40.mlcnb.errorrate.add'
'w40.mlcnb.errorrate.replace'
error rate(%)
number of principal components