The Principal Components Method as a Pre-processing Stage for Decision Tree Learning

Luboš Popelínský[1] and Pavel Brazdil[2]

[1] Faculty of Informatics, Masaryk University
Brno, Czech Republic
popel@fi.muni.cz
[2] LIACC University of Porto, Portugal
pbrazdil@ncc.up.pt

Abstract. In this paper we use the principal components method as a pre-processing stage for decision tree learners. We investigate the case when query examples are known in advance. We have evaluated an easily computable criterion for choosing the optimal number of principal components. Our experimental results show that decrease in error rate can be achieved without an increase in hypothesis complexity (the size of a decision tree). On some datasets the method competes even with C5.0 with boosting. We also show that the required time does not exceed significantly the time needed for learning a decision tree without the use of principal components. We also describe experiments with other propositional learners. Modest gains have been observed for the more common case when the query examples are not known in advance.

1 Motivation

A serious drawback of decision tree learners like C4.5 [12] is its very restricted language. In the process of knowledge discovery in databases (KDD) it is very often the pre-processing stage, namely 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 data mining algorithm. A wide range of constructive induction techniques [6, 7, 9, 15] has been explored aiming at enriching the language of propositional learners [4, 5]. The usual way is to add new attributes that are computed from existing ones using a priori given functions. One important class of functions includes linear combinations of numerical attributes. To prevent a large increase in computational complexity and to prevent over-fitting we cannot add just any linear combination. Principal components [11, 13] are useful linear functions that capture linear dependencies among attributes and hence deserve particular attention.

The method presented can be seen as a multistrategy approach [5] consisting 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 replace, or to add them to the original attributes. The first step is actually a form of constructive induction [4] applicable for numerical variables, which uses the methodology of the principal component analysis to generate derived attributes.

Principal components are of course not new. They are used sometimes when studying a particular data analysis problem [14]. They are much more widespread among statisticians and data analysts than among machine learning researchers. 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 investigate two variants of data mining. The first one concerns the case when the query examples are known in the time of learning the decision tree. The second variant is the common situation when no information about query examples is available in the moment of learning. For the former we have verified that the combination of the principal components method and decision tree learners, namely `C5.0`, achieves an increase in accuracy of decision trees and a decrease in complexity, i.e. size of a decision tree. We also show that the sum of the preprocessing time and the training time does not exceed significantly the time required for learning a decision tree without the use of principal components.

The paper is organised as follows. In Section 2 we briefly overview main ideas of principal components (PCO's). The first set of experiments with PCO's is described in Section 3. The main method is described in Section 4. Section 5 contains summary of results for six datasets examined in METAL project [1] [2]. In Section 6 we display results for the case when the query examples have been unknown in the time of learning.

2 Principal Component Method

Principal components analysis [11, 13] (or the Karhunen-Loeve expansion) is a statistical method for dimensionality reduction or exploratory data analysis. 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 [11, 13] that the principal components c_i are equal to eigenvectors of the variance matrix V.

3 Replacing versus Adding PCO's

Principal components are commonly used to *replace* attributes with smaller number of new attributes. The second way is to *add* principal components to the

[1] The datasets represent a subset of about 50 datasets used in comparative studies within project METAL.

original dataset. We have evaluated both methods on six datasets from UCI repository [3] containing only numerical attributes with no missing values. The following datasets were used:

```
satimage, german_numb, waveform21, waveform40, letter,
segment.
```

For each datasets we first replaced attributes with 1,2,..,7 principal components and then used `c50tree` (C5.0 with standard settings of parameters, without boosting, 10-cross-validation). Then we ran `c50tree` on datasets extended with the same number of PCO's.

The results obtained by adding attributes were on the whole better then those obtained by replacing attributes. For `satimage` dataset, for instance, the error rate of the decision tree was about 1% better when attributes were added than in the other case. With `waveform` datasets, the replacement worked better, but the difference was quite small. For `letter` and `segment` the error rate with replacement was much worse than for the extended dataset. To simplify matters we decided to explore just the way of adding principal components to the original set of attributes.

The main features of this method are described in the next section. Section 5 contains a summary of the results obtained when query examples have been known. Section 6 brings results for the common case – when the query examples are unknown.

4 Adding Principal Components

4.1 Optimal number of principal components

One question that needs to be answered is: what is the optimal number of PCO's that we should add to the existing attributes? 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. Another possibility is to set the number of PCO's to the number of eigenvalues that are significantly greater than one. It was shown [14] that the right approximation does not realy need to compute complex data characteristics like variances. It can be computed using the following criterion:

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

We have carried out an experimental study to examine how this criterion works in practice. The results are shown in Table 1 indicating that it works quite well. The middle part of the table contains error rates for the estimated number of PCO's using the criterion shown. For 3 out of 6 datasets the estimated number perfectly matches the ideal number of PCO's identified by an exhaustive search. For `satimage` and `waveform40` datasets the difference between error rates for the estimated number of PCO's and the ideal number was very small, see the last column in Table 1. Only for `segment` dataset the error rate was 3.6% instead of 3.12% if ideal number of PCO's were used. It means that for 5 out of 6 datasets the criterion is usable. As can be seen the error rate associated with the recommended number of PCO's is not too far from the ideal value. Later we will show that the criterion works well even for the hypothesis complexity. Fig. 1 shows the results for `satimage` data set obtained after adding 1 up to 10 principal components.

Table 1. Estimated vs. ideal number of PCO's

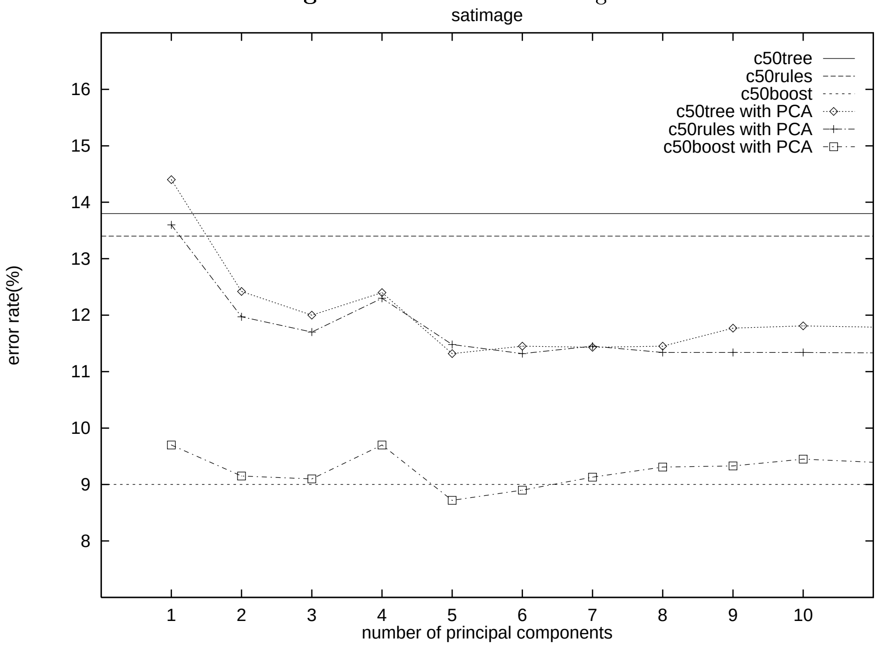
	estimated #PCO's		ideal #PCO's	
	#PCO's	error rate	#PCO's	error diff.
satimage	3	12.0	5	0.68
german_num	2	28.3	2	-
waveform21	2	17.0	2	-
waveform40	2	16.7	3	0.02
letter	4	12.0	4	-
segment	3	3.6	9	0.48

4.2 Algorithm

The method discussed earlier can be sumarized as follows:

1. Compute the value of N, the approximation to the optimal number of principal components.
2. Compute the first N principal components.
3. Add the principal components to the original dataset.
4. Run a decision tree algorithm on the extended learning set.

Fig. 1. Error rates for satimage



5 Results for known query examples

In this section, we investigate the case when query examples (actually the test set) are known in the moment of learning the decision tree. For this case, we can compute the principal components for the union of the learning and the test set together. It is obvious that we use unlabeled examples, i.e. examples without information about their class membership. We will show that the information contained in these unlabeled test examples results in the high accuracy of leraning.

Each machine learning task can be characterised by certain indicators depending on user's needs. Here we focus on three of them: *error rate on unseen examples*, *hypothesis size* and *training time*. As testing time is for `C5.0` very small in comparison to the training one, we do not take it into account. All results below are computed using 10-fold cross validation.

5.1 Error rate

We have decided to compare `c50tree+` and `c50boost+` (when the estimated number of PCO's was added) with `c50tree` and `c50boost`. The results are shown in Table 2. The +/- sign means that the error rate was smaller/greater than for the case without use of PCO's. We have not yet conducted statistical significance tests. The minimal value of the error rate obtained in our experiments for each dataset is highlighted in bold. For 5 out of 6 datasets `c50tree+` achieves lower

Table 2. c50tree and c50boost

		c50tree+	c50tree		c50boost+	c50boost
satimage	+	12.0	13.4	-	9.15	**9.0**
german_num	+	28.3	29.4	+	**26.0**	26.3
waveform21	+	17.0	23.5	+	**15.7**	17.4
waveform40	+	16.7	25.1	+	**15.0**	17.1
letter	+	12.0	12.1		4.7	**4.7**
segment		3.6	3.6	+	**1.6**	1.9

error rate than `c50tree`. For `segment` dataset the error rates are equal. Using nonparametric Wilcoxon test the difference between the error rates for c50tree+ and c50tree (columns `c50tree+` and `c50tree`) is significant on the level of 95 %. In two datasets – `waveform21` and `waveform40` – `c50tree+` is even more accurate than `c50boost`.

Similar gains can be observed if we compare `c50boost+` and `c50boost`. For 4 out of 6 datasets `c50boost+` is more accurate than the other counterpart. Only for `segment` dataset its accuracy is worse.

5.2 Hypothesis size

We were interested in comparing the sizes of decision trees with the corresponding sizes of trees when using PCO's. For 3 out of 6 datasets the size of the decision tree generated decreased when using principal components. Some small increase in hypothesis complexity was observed for `satimage` and `letter` datasets, but the difference was only 1.2% and 4.5%. The only large difference was obtained with `segment` dataset as can be seen in Table 3. The column 'PCO' in this table contains the complexity of the principal components expressed in terms of the number of coefficients (product of the number of attributes and the number of PCO's). Column 'total' contains the sum of the tree size and the number of PCO coefficients. Sign '-' means that the hypothesis size is smaller for `c50tree` with principal components. It is true that decision trees are intelligible to human unless they are very complex. On one side, after adding PCO's the tree complexity decreased enormously (see the 'tree size' column in Tab. 3) for 4 out of 6 data sets. On the other side, when adding the complexity of PCO's, this decrease has been partially negated. Another problem is that the principal components computed are difficult to interpret. It again decrease intelligibility of the resulting decision tree.

The criterion used to estimate the required number of PCO's worked well even for hypothesis size (Table 4). It worked well with 3 out of 6 datasets. In 3 datasets

Table 3. Hypothesis size

		c50tree+			c50tree
		tree size	PCO	total	
satimage	+	170.0	108	278.0	274.6
german_num	-	75.5	6	81.5	94.8
waveform21	-	120.1	42	162.1	274.7
waveform40	-	181.7	80	261.7	300.2
letter	+	1168.5	64	1232.5	1178.8
segment	+	40.3	57	97.3	42.3

Table 4. Optimal vs. best number of PCO's

	estimated #PCO's		optimal #PCO's	
	#PCO's	hypothesis size	#PCO's	size diff.
satimage	3	170.0	6	10.1
german_num	2	75.5	2	-
waveform21	2	120.1	2	-
waveform40	2	181.7	2	-
letter	4	1168.5	5	15.7
segment	3	40.3	1	0.1

a smaller hypothesis size was observed for another number of PCO's, but the difference did not exceed 2%.

5.3 Training time

In our analysis of times, 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. The results are shown in Table 5. The last column displays the ratio of the total CPU time of `c50tree+` and `c50tree`. An increase in the

Table 5. Training time

	c50tree+			c50tree	c50tree+ /
	PCO time	train. time	total	time	c50tree
satimage	2.11	3.04	5.15	4.55	1.13
waveform21	1.93	2.25	4.18	2.55	1.63
waveform40	3.75	4.65	8.40	5.37	1.56
letter	5.99	14.87	20.86	7.68	2.72
segment	0.83	0.72	1.55	0.61	2.54

required time was observed for two datasets – `letter` and `segment`. The reason may be the fact that there is no strong linear dependency among attributes.

5.4 Other learners

We also extended other propositional learners that are frequently employed in data mining, by adding the PCO pre-processing stage. The following propositional learners were used: linear discriminant method `lindiscr`, linear trees `ltree` (a decision tree that can introduce linear function in its nodes) [8] , instance-based learner `mlcib` [1] and naive Bayes classifier `mlcnb` [1, 10]. Table 6

Table 6. Error rates for other learners

	lindiscr+	ltree+	mlcib+	mlcnb+
satimage	+15.72	-12.04	-9.11	+18.92
wav21	+13.94	-15.68	+23.02	+17.78
wav40	-14.06	+15.28	+28.72	+18.48
letter	-29.80	+12.98	+ 5.80	-37.56
segment	- 8.18	+ 2.90	+ 5.54	-20.73

shows the results. The sign +/- means that the error rate when PCO's was employed was smaller/greater than the error rate for the original dataset. For all algorithms the PCO's brought advantage in some datasets. Note that the differences shown need not be statistically significant. The total time for training and PCO's was generally greater than the training time for the original version. One interesting exception is `ltree` on `satimage` data that should be investigated further.

Table 7. Training time for other learners

	lindiscr+	ltree+	mlcib+	mlcnb+
satimage	1.47	10.97	2.76	3.52
wav21	0.84	6.65	2.18	2.68
wav40	1.88	11.31	3.89	4.87
letter	2.90	420.46	7.26	10.24
segment	0.43	1.92	1.08	1.31

6 Results when queries are unknown

We tested the same method for the more common setting in data mining – when the test examples are unknown in the time of learning. 90% of randomly chosen

Table 8. Results when queries are unknown

		c50tree+	c50tree
satimage	+	13.9	14.1
german_num	+	30.8	33.2
waveform21	+	21.7	23.0
waveform40	+	30.8	33.2
letter	-	22.0	11.6
segment	-	6.3	2.9

examples were used for learning, 10% for testing. Table 8 display the average of error rates from 5 runs. We can see that for 4 out of 6 data sets the modest gains were observed. For `letter` and `segment` the error rate increased. Concerning the hypothesis size, the decision tree for data with PCO's were smaller for three data sets – `satimage, wav21, wav40`. After adding the complexity of PCO's (computed by the same way as in the previous section), only for `wav21` the hypothesis size was smaller than that one for the case without PCO's. Comparison of training times with and without PCO's did not change from that one described in the previous section.

7 Conclusion

We described a method that employes the principal components analysis as a preprocessing stage for propositional learners. We have shown that when knowing the query examples 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). Our experimental analysis has shown that the criterion for estimating the number of principal components is quite sound. We also demonstrated that for other propositional learners principal components brings some advantages, too.

To summarise our results for C5.0 decision tree learner:

– adding principal components to the original dataset results in a significant decrease in error rate without significantly increasing the required time;
– for some datasets the results even exceed `c50boost` ;
– complexity of hypothesis does not increase.

For the usual setting, i.e. when the query examples are unknown, we observed only the modest gains for 4 out of 6 data sets. We claim that the information concerning query examples is useful and should always be supplied to the system, whenever available.

Concerning future work, we need to find a criterion that for any given dataset says whether (and maybe how much) principal components will help to decrease error rate (see results for `letter` and `segment` in Tab. 2 and Tab. 8). We did not deeply addressed the problem when PCO's are used with `c50boost` (C5.0 with boosting). The results for `satimage` data (see Fig. 1) display a different trend for `c50boost+` if compared with `c50boost`, than for `c50tree+`.

It should be noted that our results should be regarded as preliminary. We have carried out the systematic comparisons on 6 datasets only. The study should be extended to more datasets. Furthermore, the method is applicable to datasets with numeric attributes. Further work should be carried out to see how the technique should be extended to cope with symbolic attributes, using for instance correspondance analysis or other suitable statistical techniques.

Acknowledgements

Thanks to Luis Torgo, Joaquim Costa, João Gama and other members of LIACC of University of Porto for their help. Comments of anonymous referees help us to improve this text. 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. UCI Machine Learning repository. `http://www.ics.uci.edu/AI/ML/Machine-Learning.html`
4. 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.
5. Bloedorn E., Wnek J., Michalski R.S.: Multistrategy Constructive Induction: AQ17-MCI. *In Michalski R.S. and Tecuci G.(eds.), Proceedings of the Second International Workshop on Multistrategy Learning (MSL-93), Harpers Ferry, W.VA.*, pp.188-206, 1993.
6. Bloedorn, E., Michalski, R.S. and Wnek, J., "Matching Methods with Problems: A Comparative Analysis of Constructive Induction Approaches," Reports of the Machine Learning and Inference Laboratory, MLI 94-2, School of Information Technology and Engineering, George Mason University, Fairfax, VA, May 1994.
7. 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.
8. Gama J.: Probabilistic linear tree. *In D. Fisher (ed.), Proceedings of the 14th Int. Conf. on Machine Learning (ICML'97)*, Morgan Kaufman, 1997.
9. Michalski, R.S.: Inferential Theory of Learning as a Conceptual Basis for multistrategy Learning. *Machine Learning*(Special Issue on Multistrategy Learning) 11, 1993
10. Mitchell, T.M.: *Machine Learning.* McGraw Hill, New York, 1997.

11. F. Murtagh and A. Heck: *Multivariate Data Analysis.* Kluwer Academic, Dordrecht, 1987.
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. 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.