IMPLEMENTING DECISIONS IN BINARY DECISION TREES USING INDEPENDENT COMPONENT ANALYSIS

Petteri Pajunen

Helsinki University of Technology
Lab. of Computer and Information Science
P.O. Box 5400
FIN-02015 HUT
Finland

Mark Girolami

University of Paisley
Dept. of Computing and Information Systems
Paisley, PA1 2BE
Scotland

ABSTRACT

There are various ways to implement decisions in binary decision trees. Most approaches can be interpreted as greedy methods optimizing some local goodness criteria at each node. Often it is required that either target values (regression trees) or class labels (classification trees) are available. In this paper linear ICA is applied to implement the decisions in a binary tree. The rationale is that ICA can be used to find directions where the data has "structure". The linear transformation defined by ICA can then be interpreted as a change of variables, where the new variable captures the structure, e.g. has smallest entropy. The linear decisions are then made by tresholding the variable. An experiment is presented which shows that the proposed method can find reasonable representations of real-world data in an unsupervised manner, i.e. without using class labels.

1. INTRODUCTION

Decision trees are a common methodology for inductive inference. There benefits include fast computational properties and a representation that may be easy to interpret in terms of decisions. Typically they are used when data is discrete-valued but extensions to continuous-valued data exist.

Decision trees can be used for solving classification and regression problems. Especially binary trees provide a computationally effective solution, since only $\log_2 n$ decisions have to be made for a tree with n leaves. Furthermore, if the decisions are linear, then each decision is essentially an inner product between a data vector and a normal vector of a separating hyperplane.

A binary decision tree works by recursively partitioning the data set into two. This series of decisions implicitly defines a binary tree, where the decisions are made at the nodes. Depending on the result of each decision, the observed sample continues down the tree to a child node or leaf. Finally, each observation ends up in a leaf and hence the tree has classified the data.

The typical problem where decision trees are applied is when the observed data is as set of multivariate attribute values, e.g. $\mathbf{x}(k) = [age(k), sex(k), height(k)]$. In basic methods, the decisions are made by choosing one of the attributes and computing a threshold, which then implements a binary decision on all observations. For example, we might choose the attribute $sex(k)$ as the deciding attribute and then separate the observations into classes *male* and *female*.

The choice of the attribute in the ID3 algorithm [1] is based on the concept of *information gain*. The attribute which reduces entropy the most is selected for implementing the decision. This measure, however, depends on using training data in which we know the correct classes for each observation. The information gain then chooses the decision that gives the best partitioning with respect to classification performance. The algorithm can be therefore interpreted as a greedy optimization method for minimizing classification error using a binary tree.

2. DECISION TREES WITH LINEAR COMBINATIONS OF CONTINUOUS-VALUED ATTRIBUTES

The basic approaches, e.g. ID3 have been extended in various directions. Continuous-valued attributes have been considered in [2, 3] and decisions made by thresholding linear combinations of continuous-valued attributes in [4, 5]. However, the decisions are made by depending on known class labels for the observations. In this paper we consider implementing the decisions by thresholding linear combinations without depending on class labels, i.e. we depend alone on the data distribution

$p(\mathbf{x})$.

3. LINEAR DECISIONS AND INDEPENDENT COMPONENT ANALYSIS

If a multivariate data set is considered in the classification problem, each decision divides the data set in two. Intuitively, it is desirable to divide the data set so that different types of samples go to different parts of the tree. With known class labels this can be achieved using one of various information gain measures, or simply by computing the optimal separating hyperplane. However, it is our purpose to construct a decision tree without knowing the class labels.

In this paper, a single ICA vector is used to compute the decision rule in each node. The ICA vector $\mathbf{w}$ defines the direction in which the data set has most "structure" as explained below. The decision rule is then implemented as a separating hyperplane with normal $\mathbf{w}$.

In Independent Component Analysis it is normally assumed that the observed multivariate data is generated by the generative statistical model

$$\mathbf{x}_n = \mathbf{A}\mathbf{s}_n + \epsilon_n$$

where the components of $\mathbf{s}_n$ have some desirable properties to facilitate solving the mixing matrix $\mathbf{A}$. However, finding the inverse of $\mathbf{A}$ involves in theory the minimization of mutual information [6, 7, 8, 9] which can be decomposed as

$$I(\mathbf{y}) = \log|\mathbf{A}| + \sum_i H(y_i) + H(\mathbf{x}).$$

Therefore ICA is often solved one component y_{in} at a time by imposing certain restrictions on the separating matrix. Consequently, one is looking for directions that *minimize entropy.* A single row vector of the separating matrix $\mathbf{W}$ gives one component $y_n = \mathbf{w}^T\mathbf{x}_n$.

Entropy can be interpreted as a measure of structure in the data if we accept the notion that non-Gaussian distributions imply more structure than Gaussian distributions. This can be demonstrated for example considering multimodal distributions which have clearly non-Gaussian distributions when we consider directions which would retain the multimodal distribution in projections. Other directions where the different modes would overlap in projections are closer to Gaussian and simultaneously do not have the visible multimodal structure. For exploratory data analysis looking for such directions have been proposed in [10].

From the classification point of view, we would like to implement a decision that effectively separates different types of data samples to different outcomes of the decision. Looking to implement a linear binary decision means that we need to define a hyperplane which separates the samples into two classes. If we choose this hyperplane as the one defined by $\mathbf{w}$ as the normal vector, then the hyperplane will be orthogonal to the most interesting direction.

This can be interpreted in another way: the linear transformation $y_n = \mathbf{w}^T\mathbf{x}_n$ is a change of variables into a new variable which has most structure. The decision rule is then simply a thresholding operation on the new variable.

4. A PROBABILISTIC INTERPRETATION

A soft binary partitioning at each node can be achieved by considering the following log-likelood expression which assumes that the priors are both equal.

$$\mathcal{L} = \sum_{n=1}^{N} log \sum_{k\in\{C_1,\, C_2\}} p(y_n|C_k, \mathbf{w})/2$$

Considering the binary partition as being composed of two equiprobable zero-mean and unit variance Gaussians as in [11, 12] gives the following gradient term.

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \sum_n \{2P(C_1|y_n) - y_n - 1\}\,\mathbf{x}_n$$

Where $2P(C_1|y_n) = 1 + \tanh(\mathbf{w}^T\mathbf{x}_n)$. Now the EM algorithm can be applied to solve this system of equations by computing the posterior probability of the dichotomy in the E-step and re-estimating the vector $\mathbf{w}$ in the M-step. Setting the gradient of the log-likelihood to zero yields

$$2\sum_n P(C_1|y_n)\mathbf{x}_n = \sum_n \mathbf{x}_n + \sum_n \mathbf{x}_n\mathbf{x}_n^T\mathbf{w}$$

and noting that the sample mean is zero gives the following M-step.

$$\mathbf{w}^{new} = 2\mathbf{R}_{\mathbf{xx}}^{-1}\langle\mathbf{x}\rangle$$

Where $\mathbf{R}_{\mathbf{xx}}$ is the sample covariance matrix of the data allocated to each node and $\langle\mathbf{x}\rangle$ denotes the posterior class mean estimate $\sum_n P(C_1|y_n)\mathbf{x}_n$ This can be further simplified by *pre-whitening* the data at each node thus $\mathbf{R}_{\mathbf{xx}} = \mathbf{I}$ and so the EM update boils down to a batch form of a one-unit Hebbian update term $\mathbf{w}^{new} = \sum_n \tanh(\mathbf{w}^{old}\mathbf{x}_n^T)\mathbf{x}_n$.

Of course an instantaneous gradient update could also be used and again noting that the observations are zero-mean

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}}\mathbf{w}^T\mathbf{w} = \sum_n \{2P(C_1|y_n) - y_n\}\, y_n\mathbf{w}$$

follows the relative gradient update which was used in multivariate form for hierarchic data clustering in [11].

$$\Delta \mathbf{w} = \mu(1 + \tanh(y)y - y^2)\mathbf{w}$$

5. CONSTRUCTING THE BINARY DECISION TREE

The available training data $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}$ can be used to implement a binary decision tree as follows:

1. estimate the sample mean of the observations $\mathbf{x}_n$ and transform by subtracting this estimate: $\mathbf{x}'_n = \mathbf{x}_n - \hat{E}\{x_n\}$
2. use the zero-mean observations to find an ICA vector $\mathbf{w}$ yielding the structured variable $y_n = \mathbf{w}^T\mathbf{x}'_n$
3. partition the observations into two classes:

$$\mathbf{x}'_n \in \begin{cases} \text{Class 1}, & \mathbf{w}^T\mathbf{x}'_n \geq 0 \\ \text{Class 2}, & \mathbf{w}^T\mathbf{x}'_n < 0 \end{cases}$$

   or using a soft partition then

$$\mathbf{x}'_n \in \begin{cases} \text{Class 1}, & P(C_1|\mathbf{x}'_n) > P(C_2|\mathbf{x}'_n) \\ \text{Class 2}, & \text{otherwise} \end{cases}$$

   where the posteriors are defined above.

4. repeat the above steps for both subclasses

The construction of the binary tree is limited by the number of observed samples since reliable estimation of ICA requires a sufficient amount of data. However, this limitation is somewhat less severe compared to regular ICA since only one component is computed at each node.

6. SELECTING THE SIZE OF THE DECISION TREE

The above method for constructing the decision tree did not include any methods for model selection, i.e. deciding the size of the tree. Growing the tree as large as possible suffers from *overfitting* which means that properties of the training data are being modeled that do not generalize to unobserved data.

There are two general ways to limit the size of the decision tree:

- stopping rules, which prevent further splitting of the nodes
- pruning rules, which discard decisions after the tree has been constructed

Most of the methods require that the class labels are known. For example using validation sets can be used to test the usefulness of further splitting. If the performance increases on the validation set, then the new decision is included.

For our approach, the most promising approach is to use information-theoretic criteria such as MDL [13, 14] to measure if the decision wastes more information than it saves [15]. This has not yet been implemented in our approach and it is considered further research. However in the experimental work reported we note that the transformed variables will have a non-Gaussian structure typified by the two class modes of the binary partition. It is therefore reasonable to exploit a measure of non-Gaussianity of the projected data; nodes whose transformed variables exhibit Gaussian characteristics[1] can therefore be identified as leaf nodes.

7. ICA DECISION TREES IN CLASSIFICATION

A simple way to perform a classification task using binary trees is to construct the tree first without regarding class membership of the observations. Afterwards, each leaf of the tree may be labeled by estimating the conditional class membership probabilities. The leaf node can then be classified by choosing the most probable class as the correct label.

In machine learning literature binary decision trees have been proposed which construct linear decisions at tree nodes using information about class membership of the samples. The proposed method here applying independent component analysis differs mainly in its selection of the linear decisions: only the underlying probability distribution $p(\mathbf{x})$ of the multivariate observations is used. The approach is useful also when class memberships of the observations are unknown

8. EXPERIMENTS

To investigate the ability of ICA decision trees to represent clustered data, we chose a data set with known class labels and applied it to learn a decision tree. However, the known class labels were ignored in learning the tree. The leaves were assigned class labels only after the linear decisions had been fixed.

The data applied was the Oil Pipeline Data used in [11, 16]. The training data consists of twelve-dimensional samples each of which is classified into one of three possible classes.

[1]Uni-modalilty was considered as a reasonable indicator for terminal nodes

The training set contained 1000 samples, which makes it possible to use a binary tree with depth at most 10. The depth must be somewhat smaller since there must be enough training samples at each node where the linear decisions must be computed.

First, all samples were assigned to the root of the tree and after removing the mean value a single ICA vector was computed. For these experiments the algorithm used was the relative gradient rule [11, 17, 18]

$$\Delta \mathbf{w} = \mu(1 + \tanh(y)y - y^2)\mathbf{w}$$

The choice of nonlinearity in the above rule is justified in [11] where it is derived from the symmetric Pearson mixture model. The discussion in the section(4) motivates the use of such an ICA rule for binary classification at each node.

After learning the vector $\mathbf{w}$ each zero-mean sample $\mathbf{x}_n$ was classified according to the sign of the inner product $\mathbf{w}^T\mathbf{x}_n$. This is simply the projection of $\mathbf{x}_n$ in the direction defined by the ICA vector $\mathbf{w}$. Another interpretation is that the decision is made by a thresholding operation of the variable $y_n = \mathbf{w}^T\mathbf{x}_n$ which could be based on the maximum estimated class posterior probability.

The results of the linear decision rule partition the training samples into two subsets. Each of these sets is assigned to the left and right children (nodes immediately below) of the current node. The above procedure is recursively called at each node of the tree, with the exception of the leaf nodes.

In the reported experiment, the tree was constructed without taking into consideration the known class labels. Only the leaf nodes were assigned a class label by selecting the most probable class. This was done by choosing the class with most training samples allocated to the leaf node.

A test set of 1000 samples was used to measure the performance of the decision tree in terms of classification error. The result was a 78% accuracy measured in the test set. The classification accuracy measured in the training set was 88%. Although it is not meaningful in general to measure classification performance in the training set, in this case it illustrates how well the proposed algorithm is able to represent the classes in the data. Remember that the representation was computed *without* using the class labels. Good classification performance in the test set implies that mostly samples from the same class end up in any one of the leaf nodes.

Unsupervised learning, however, cannot be expected to match the performace of classifiers, which take into consideration the known class labels of training data. As a reference figure, a multi-layer perceptron trained using the class labels achieved a classification error of 98.6% on a test set of this data [16].

9. SUMMARY AND CONCLUSIONS

Benefits of decision trees include quick optimization due to greedy approach of recursively selecting the suboptimal decisions and fast computation of the decisions in applying the tree. These benefits are not dependent on knowing the class labels of the observations and therefore can be utilized in knowledge discovery.

Similarly, Independent component analysis requires no class labels to represent data but enjoys certain information-theoretic properties that suggest its applicability in data analysis. An especially simple way of using ICA is to compute a single ICA vector and interpret it as the most interesting direction as in projection pursuit.

In this paper, we combined the benefits of decision trees and ICA to implement a linear decision based on an attribute representing the interesting direction. It was found that the resulting decision trees are fast to compute and seem to represent data well without using class labels in training.

Further research includes investigating information-theoretic stopping criteria to prevent overfitting and combining ICA and class labels in implementing decisions.

10. REFERENCES

[1] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[2] U. Fayyad, *On the Induction of Decision Trees for Multiple Concept Learning.* PhD thesis, University of Michigan, EECS Department, 1991.

[3] U. Fayyad and K. Irani, "Multi-interval discretization of continuos-valued attributes for classification learning," in *Proceedings of 13th International Joint Conference on Artificial Intelligence* (R. Bajcsy, ed.), pp. 1022–1027, Morgan-Kaufmann, 1993.

[4] P. Utgoff and C. Brodley, "Linear machine decision trees," Tech. Rep. COINS 91-10, University of Massachusetts, Amherst, MA, USA, 1991.

[5] S. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *Journal of Artificial Intelligence Research*, vol. 2, pp. 1–33, 1994.

[6] P. Comon, “Independent component analysis – a new concept?,” *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[7] H. Yang and S. Amari, “Adaptive online learning algorithms for blind source separation: Maximum entropy and minimum mutual information,” *Neural Computation*, vol. 9, no. 7, pp. 1457–1482, 1997.

[8] P. Pajunen, “Blind source separation using algorithmic information theory,” *Neurocomputing*, vol. 22, pp. 35–48, 1998.

[9] P. Pajunen, “Blind source separation of natural signals based on approximate complexity minimization,” in *Proc. Int. Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, (Aussois, France), pp. 267–270, January 1999.

[10] J. Friedman and J. Tukey, “A projection pursuit algorithm for exploratory data analysis,” *IEEE Tr. on Computers*, no. 23, pp. 881–889, 1974. Ser. C.

[11] M. Girolami, “Hierarchic dichotomizing of polychotomous data - an ICA based data mining tool,” in *Proc. Int. Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, (Aussois, France), pp. 197–202, January 1999.

[12] M. Girolami, A. Cichocki, and S. Amari, “A common neural network model for exploratory data analysis and independent component analysis,” *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1495–1501, 1998.

[13] J. Rissanen, “Modeling by shortest data description,” *Automatica*, vol. 14, pp. 465–471, 1978.

[14] G. Schwarz, “Estimating the dimension of a model,” *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[15] M. Mehta, J. Rissanen, and R. Agrawal, “MDL-based decision tree pruning,” in *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, (Menlo Park, CA), pp. 216–221, AAAI Press, 1995.

[16] C. Bishop and G. James, “Analysis of multiphase flows using dual-energy gamma densitometry and neural networks,” *Nuclear Instruments and Methods in Physics Research*, no. A327, pp. 580–593, 1993.

[17] S. Amari, “Natural gradient works efficiently in learning,” *Neural Computation*, vol. 10, pp. 251–276, 1998.

[18] J.-F. Cardoso and B. Laheld, “Equivariant adaptive source separation,” *IEEE Trans. on Signal Processing*, vol. 44, no. 12, pp. 3017–3030, 1996.