BICA: a Boolean Independent Component Analysis Algorithm

Bruno Apolloni, Dario Malchiodi
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico 39/41, 20135 Milano, Italy
{apolloni, malchiodi}@dsi.unimi.it

Andrea Brega
Dipartimento di Matematica "F. Enriques"
Università degli Studi di Milano
Via Saldini 50, 20133 Milano, Italy
brega@dsi.unimi.it

Abstract

We introduce a procedure for mapping general data records onto Boolean vectors, in the philosophy of ICA procedures. The task is demanded of a neural network with double duty: i) extracting a compressed version of the data in a tight hidden layer of a self-associative multilayer architecture, and ii) mapping it onto Boolean vectors that optimize an entropic target. We prove that the components of these vectors are approximately independent and appreciate their ability to preserve data information in a statistically driven solution of benchmark classification problems.

1. The BICA algorithm

A suitable way of taking decisions based on data is to split the decision process in two steps. The first is devoted to mapping original variables onto propositional variables. The second step combines them into a suitable Boolean function [1]. The benefit of this procedure is twofold.

- From a strictly operational point of view, it depends on the ability to rely on short formulas that describe this function, which stands for its easy usability and understandability. This calls, among other things, for efficient compression of the data into Boolean vectors to be considered as assignments to the above propositional variables.

- From a cognitive perspective, we may map the two phases onto the environment-representation and rule-identification tasks of a cognitive system. The former is a key step in enabling the cognitive system to behave as a truly autonomous agent. As a matter of fact, its autonomy resides mostly in the ability to build its own identification and representation of the environments it is embodied into [4].

Cognitive perspective is highly compliant with the strategy we use for compressing data. Consider a multi-agent system (MAS) [9] where artificial agents are called on to cooperate in achieving a task. The first thing they must do is communicate their goals with each other. Now, if their ontologies [8] are completely unknown to each other, they must discover *ex novo* a common vocabulary for interpreting the signals they exchange whose sole feedback is the suitability of the joint actions they perform. We too, in our approach, do not bother with the semantics of the symbols we produce in the first step, because we expect that it will emerge from the statistics on their use. Rather, we are looking for an actual random mapping with some semantic and entropic constraints that come from an efficient use of the new variables precisely for taking decisions (which we assume here to be binary) based on them. Namely, we look for a vector of Boolean variables $\mathbf{v}$ whose assignments reflect (possibly through a distorting mirror) the relevant features of the original – possibly continuous – data pattern $\mathbf{x}$. This means that a correspondence must exist between the two parts, where Boolean assignments may coincide when they code data patterns having the same value of a $0/1$ decision variable. It is precisely to improve the decision's correctness for new patterns that we need the mapping in order to compute independent Boolean variables. This seems like a typical task for unsupervised *vector quantization* methods [7]. But we want to avoid unnecessary topological clustering constraints at the basis of competitive methods such as *self organizing maps* [11] or *radial basis functions* [10]. In summary, on the one hand, we want to extract independent components of the signals, as the *noble part* of their information content. On the other hand, stressing the fact that independence is a property of the representation of the data that we use, we search for this property precisely on a Boolean representation of them suitable for correctly partitioning them into positive and negative inputs of our decision rule. Accordingly, we call our method Boolean Independent Component Analysis, BICA for short.

Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS'05)
0-7695-2457-5/05 $20.00 © 2005 IEEE

1.1. The architecture

We split the mirroring of the original data into the target Boolean vector in two parts: a true mirroring of the patterns and a projection of a compressed representation of them (obtained as an aside result of the first part) into the space of Boolean assignments. The whole process is done by a neural network with an architecture shown in Fig. 1 sharing the same input and hidden layer with the two output segments A and B computing the Boolean assignments and a copy of the input, respectively.

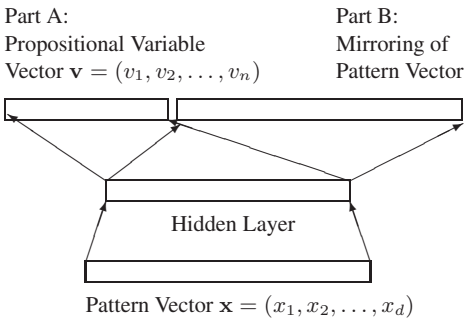

Figure 1. Layout of the neural network mapping features to symbols.

1.2. The learning algoritm

We train this network with a backpropagation algorithm [18] whose δ function is specified as follows.

Error backpropagation in part B Mirroring is a usual functional requirement for an MLP [14]. We structured our network as a three-layer network with the same number of units in both input and output layers and a smaller number of units in the hidden layer. Therefore the hidden layer constitutes a bottleneck which collects in the state of its nodes a compressed representation of the input. This part of the network is trained according to a quadratic error function. Hence the error $\delta_{s,j}$ which is backpropagated to the hidden layer from each unit j of this part upon presentation of s-th input pattern is:

$$\delta_{s,j} = f'_{\text{act}}(\text{net}_{s,j})(x_{s,j} - z_{s,j}) \tag{1}$$

where $\text{net}_{s,j}$ is a weighted sum of the inputs to the unit j upon presentation of s-th pattern suitably normalized, $z_{s,j}$ the corresponding output, and f_{act} is the sigmoid function [3].

Error backpropagation in part A Things are different for the units of part A of the output. In this case we require that

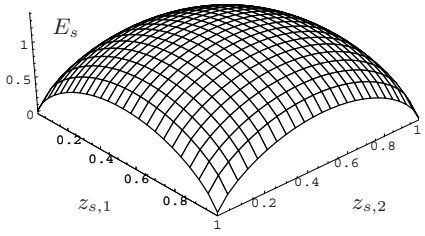

Figure 2. Graph of the function E_s with $n = 2$.

the network minimizes the following error:

$$E_s = \ln\left(\prod_{k=1}^{n} z_{s,k}^{-z_{s,k}}(1 - z_{s,k})^{-(1-z_{s,k})}\right) \tag{2}$$

This function, which we call the *edge pulling function*, has the shape of an entropy measure that finds its minima in the vertices of the neural network output space (see Fig. 2). The error which is backpropagated from the units of part A is:

$$\delta_{s,k} = f'_{\text{act}}(\text{net}_{s,k})\alpha_{s,k} \tag{3}$$

where

$$\alpha_{s,k} = -\frac{\partial E_s}{\partial z_{s,k}} = \ln\left(\frac{z_{s,k}}{1 - z_{s,k}}\right) \tag{4}$$

1.3. Directing the mapping

Using $\alpha_{s,k}$ as in (4), we let the network decide independently about the values of part A to which it will converge for each input pattern. Of course, parameters such as the learning rate, the initial weights and the influence from part B (through the hidden nodes) play an important role in this decision and constitute at the same time the source of randomness of our compression. But we want to govern this sub-symbolic process also with syntactic feedbacks that we distinguish in two kinds: local and global. At a local level, the general idea is to insert into the α expression an extra term which has the form of 'directed noise' added to the initial value of α when we are not satisfied with the 'correctness' of the result. Effectively, when the convergence value for some unit is not satisfactory we 'shake' the network in order to search for a new equilibrium. Namely, we relate this 'punishment' action with the Hamming distances of the formed propositional vectors in order to avoid inconsistency. We want to force different vectors, with a Hamming distance over a given threshold, in correspondence to patterns that we know to belong to different categories of our interest. Namely, we introduce a punishment action that for an incorrect pattern s is set to $\tau_{s,k}$ either uniformly on all output nodes or on a subset of those non contributing

to the Hamming distance increase. Its value contributes to $\alpha_{s,k}$ with the following function $\theta_{s,k}$:

$$\theta_{s,k} = (1 - 2\Gamma(z_{s,k}))\,\tau_{s,k} \tag{5}$$

where Γ is a threshold function. The first term in the brackets specifies the sign of $\theta_{s,k}$ so that the contribution to the network parameters is in the opposite direction from the one the unit is moving in. Finally, using a tuning parameter π_A to balance the mutual relevance of corrections coming from parts B and A, we get the complete expression of $\alpha_{s,k}$ which reads:

$$\alpha_{s,k} = \pi_A \left(\theta_{s,k} + \ln\left(\frac{z_{s,k}}{1 - z_{s,k}}\right)\right) \tag{6}$$

1.4. In search of independent components

The joint goal of diminishing E_s and maintaining the patterns well separated into positive (label 1) and negative ones (label 0) brings the Boolean assignments to figure as samples of independent random variables, thus we may say that these variables are *expectedly* independent.

Lemma 1.1. *With reference to the neural network and training algorithm described above, if the neural network outputs are correct and all close to the vertices of the Boolean hypercube then their values stretched to the vertices constitute assignments to expectedly independent Boolean variables.*

Proof. Via Jensen inequality [16] on the function $g(x) = x \ln x$ we prove that the normalized sum $\breve{H}$ of E_s over the patterns is majorized by the empirical entropy $\widehat{H}$ of the joint distribution of the propositional variables Z_k [1], when they are supposed to be Bernoulli i.i.d. In formulas:

$$\breve{H} \equiv \frac{1}{m}\sum_s E_s \leq \sum_k \left[-\sum_s \frac{z_{s,k}}{m} \ln\left(\sum_s \frac{z_{s,k}}{m}\right) - \left(1 - \sum_s \frac{z_{s,k}}{m}\right) \ln\left(1 - \sum_s \frac{z_{s,k}}{m}\right)\right] \equiv \widehat{H}(\mathbf{Z}) \tag{7}$$

The left part of (7) has a minimum when each $z_{s,k}$ is Boolean. The right part has a minimum when the assignments are pushed towards independent Z_k's, still preserving the entropy of the features we are observing. Namely, denoting by L the Bernoulli variable recording the training set labels, and by $H(X)$, $H(X/Y)$ and $I(X,Y)$ the entropy of X, the conditional entropy of X given Y and the mutual information between X and Y [5] respectively, by definition $I(\mathbf{Z}, L) = H(L) - H(L/\mathbf{Z})$. Moreover, apart from rare pathologies the maximization of $I(\mathbf{Z}, L)$ leads to the computation of independent components Z_k of the feature vector (which stands for a reduction of their *redundancy*), see [2]. Now, $H(L)$ is independent of the coding of the patterns. Thus we aim to decrease $H(L/\mathbf{Z})$ by decreasing $\widehat{H}(\mathbf{Z})$ with the additional constraint of separating the codes of positive patterns from those of negative patterns. Indeed

- $\widehat{H}(\mathbf{Z})$ overestimates $H(\mathbf{Z})$ by a positive term accounting for the mutual information between Z_k components, that vanishes for Z_k's independent;
- we have

$$\begin{aligned} H(\mathbf{Z}) &= -\sum_k p_k \ln p_k = -\sum_{i\in A} p_i \ln p_i - \sum_{i\in\overline{A}} p_i \ln p_i \\ &= -\left(\sum_{i\in A} p_i\right) \sum_{i\in A} \frac{p_i}{(\sum_{i\in A} p_i)} \ln\left(\frac{p_i}{\sum_{i\in A} p_i} \sum_{i\in A} p_i\right) \\ &\quad - \left(\sum_{i\in\overline{A}} p_i\right) \sum_{i\in\overline{A}} \frac{p_i}{\sum_{i\in\overline{A}} p_i} \ln\left(\frac{p_i}{\sum_{i\in\overline{A}} p_i} \sum_{i\in\overline{A}} p_i\right) \\ &= -p_A \sum_{i\in A} p_{i/A} \ln p_{i/A} - p_{\overline{A}} \sum_{i\in\overline{A}} p_{i/\overline{A}} \ln p_{i/\overline{A}} \\ &\quad - p_A \ln p_A - p_{\overline{A}} \ln p_{\overline{A}} = H(\mathbf{Z}/L) + H(L) \end{aligned} \tag{8}$$

  where A is the set of positive pattern indices, $\overline{A}$ its complement, and p_x is the probability of the pattern or set denoted by the index x. Hence a reduction of $H(\mathbf{Z})$ induces a reduction of $H(\mathbf{Z}/L)$ for a correct labeling of the coded patterns (hence constant $H(L)$), which might occur when the codes are well separated;
- $H(\mathbf{Z}/L)$ and $H(L/\mathbf{Z})$ jointly decrease with an increasing correlation between L and $\mathbf{Z}$ induced by a reduction of $H(\mathbf{Z})$.

Moreover, for $z_{s,k}$ close to either 0 or 1 and binary vectors almost orthogonal (so that also $\sum_s \frac{z_{s,k}}{m}$ is close to 0), $g(x)$ behaves in (7) almost linearly, making the inequality almost an equality. Thus, within this range of values, i.e. when the network is well trained, the cost function comes close to promoting the extraction of Boolean independent components from the original data. □

Remark 1.1. *The key points of the above lemma is an almost trivial consideration that, since*

$$H(\mathbf{Z}, L) = H(\mathbf{Z}/L) + H(L) \tag{9}$$

we obtain (8) if the partitioning of the data patterns' space through $\mathbf{z}$ *is correct – in the sense that patterns mapping into a same assignment of* $\mathbf{z}$ *have a same label. In this case indeed,* $H(\mathbf{Z}, L) = H(\mathbf{Z})$. Vice versa*, a mapping from* $\{\mathbf{x}\}$ *to* $\{\mathbf{z}\}$ *brings to independent components depending on the functional definition of* $\mathbf{z}$*, i.e. on the use we will do of it.*

[1] By default capital letters (such as X, Z) will denote random variables and small letters (x,z) their corresponding realizations.

Remark 1.2. *The assumptions in the lemma hide the strong condition that the patterns are well separated by the neural network, which stands for a good generalization capability of the trained network, provided that it performs well on the training set. In an approximate satisfaction of this constraint, the claim of the lemma gives an operational way of fitting the usual goal of a clustering: great distance (in a proper metrics) between clusters, and small distance – hence minimum entropy – inside them. Actually the task we require for the network is less hard than to correctly classify the input patterns – whose success would vanish the subsequent rule building phase. We just require gathering, through their Boolean assignments, the patterns in groups that are not labeled by the network, but in own turn do not contradict the labels, such as they obviously do not for instance groups constituted by a single pattern each.*

2. StatEx learning algorithm

The variables we produce are optimized in function of their subsequent use. Hence to check the method we also considered the second step of the decision procedure. Namely, we refer to binary classification (i.e. decision) whose solution is represented in terms of Disjunctive Normal Forms (DNF). Once the original input has been binary coded as described before, our problem consists of inferring these formulas from labeled examples $\{s\}$, each consisting of a Boolean vector (our code) plus a bit saying whether it corresponds to as "yes" (label = 1) or "not" instance (label = 0) of the problem. Notwithstanding the sophisticated learning algorithms proposed in the literature for solving this problem, we set up a very essential algorithm, that we call StatEx, in order to remove the value of the inferring algorithm from the considerations we will do on the efficiency of the compression algorithm. StatEx infers sequentially the DNF monomials $v_{j_1} \dots v_{j_k}$, on a set $\{v_1, \dots, v_n\}$ of propositional variables, for any n and any k, mainly on the basis of the frequencies with which the single variables take value 1 in the positive examples. Identifying a monomial with the set of its literals, this algorithm is characterized by two steps: a forward one in which, starting from the empty monomial, it builds a new monomial adding literals to it, and a backward one in which it simplifies the monomial removing not necessary literals.

For uniformity reasons we duplicate and complement the assignments $\mathbf{v}$ in each example setting $v_{n+k} = 1 - v_k$. Accordingly we work with monotone monomials on $2n$ propositional variables, with the understatement that v_{n+k} has to be read the negated of v_k in the original set of variables. So, we exclusively work with monotone monomials.

Given a set of positive examples, in the forward phase the algorithm iteratively adds left to right to the set of the monomial literals (initially empty) the one matching the maximum number of 1 assignments in the examples until all variables are added. Given the set of negative examples, in the backward step the algorithm removes right to left all literals that leave the monomial consistent with the negative examples (i.e. that are not satisfied by any of them).

After the backward step, every positive example that is verified by the already built monomial is removed from the set of the positive examples and, until this set is empty, the algorithm restarts building a new monomial.

3. Numerical results

We compare the parameters obtained with our method with those deriving from the methods used in the literature for processing the same benchmarks. This brings us to contrast BICA with

- the well known C4.5 (ID3 in some cases) method [15], where a decision tree in terms of IF-THEN-ELSE rules is drawn directly by iterated partitioning of the sampled data on the basis of mutually exclusive tests on their range.
- the Hamming Clustering (HC) [12], that computes the shortest possible Boolean formulas describing with no misclassifications patterns seen in the training phase.
- a multilayer perceptron trained with backpropagation method (NN), to show a performance comparison with subsymbolic algorithms even if no rules are supplied.

Results in the literature concern exclusively the accuracy of the classification methods, apart HC that consider also the length of the formulas. Moreover they often report only one value for the considered parameters (presumably the best one obtained with the method). We write in bold the original values and fill up the empty cells by repeating the experiment by ourselves. In case the training set and test set are predefined and unique, and the procedure is deterministic (NN and C4.5), we proceed in a conservative way, by coupling to the possibly optimized accuracy a formula length unconstrained by the above optimization. When training and testing sets are not fixed we adopt usual random partitioning schemes of the database in order to capture the statistical behavior of the methods through mean and standard deviation of the results. We obtain these parameters still in case of single pair of sets when the procedure is stochastic (BICA and HC) by rerunning more times the procedure. The length of the *signals* processed by BICA is in bits. Still in a conservative way, we conventionally attribute a length of 4 bits to continuous variables to account the lengths of the data to be compared with those compressed by BICA. We assume indeed that these bits are sufficient to the considered methods to discriminate the data w.r.t. the classification problem they are questioned on. We compute the

	BICA					Neural Network					C4.5					Hamming Clustering				
	Length			Correct %		Length			Correct %		Length			Correct %		Length			Correct %	
	Signal	Formula				Signal	Formula				Signal	Formula				Signal	Formula			
		μ	σ	μ	σ		μ	σ	μ	σ		μ	σ	μ	σ		μ	σ	μ	σ
Breast	11	11.4	2.6	91.7	1.7						36	15.3	3.5	97.9	0.6	81	133.4	39.7	93.3	1.3
Sonar	50	101.2	9.8	75.3	19.9	240	n.a.	n.a.	**84.7**	**5.7**	240	27.9	2.3	72.1	9.7					
Iono	42	40.8	7.3	84.8	6.1						136	28	n.a.	**94**	n.a.					
Monk 1	9	83.6	8.3	68.1	1.3	24	n.a.	n.a.	**100**	n.a.	24	15	n.a.	**98.6**	n.a.	17	7	0	100	0
Monk 2	9	130	7.6	68.5	1.5	24	n.a.	n.a.	**100**	n.a.	24	47	n.a.	**67.9**	n.a.	17	249.38	4.81	75.5	1
Monk 3	9	49.5	5.6	79	1.3	24	n.a.	n.a.	**93.1**	n.a.	24	9	n.a.	**97.4**	n.a.	17	67.5	6.3	92.8	1.31

Table 1. BICA experimental profile in comparison with other information management methods. μ : mean value when σ is available, single trial value when σ is not available (n.a.) σ: standard deviation. Correct%: % of correctly classified patterns of the test set.

length of a formula as the number of literals involved in. Moreover, we binarize the non binary antecedents occurring with C4.5, such as "$0.3 \leq x_i \leq 0.7$" in conjunction with "$0.5 \leq x_i \leq 0.8$" by introducing dummy variables.

The choice of a suitable size of the Boolean vector is done by trials. The entire procedure lasts a few seconds on a standard PC.

3.1 Wisconsin Breast Cancer database

It is a collection of 699 patterns of 9 discrete features to be used for cancer diagnosis. 458 patterns belong to the "benign" class, and the remaining to the "malignant" class. The features take values from 1 to 10, and refer to: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. A multiple hold out scheme is realized through 50 replicas of training and testing sets obtained by randomly splitting the databench into 372 and 311 patterns respectively [19], while 16 patterns with empty fields have been deleted. C4.5 uses data as they are. Hamming Clustering algorithm uses a a *unitary encoding* of 9 bits.

BICA reduces the lengths of both data representation and formulas description w.r.t. both reference methods at the expenses of a few percentage points loss in accuracy. Literature values with HC are length $= \mathbf{130}$, correct $= \mathbf{95.5}$ [12].

3.2 Sonar database

The goal is to discriminate sonar signals denoting a mine among 208 patterns, each made up of 60 continuous features between 0 and 1. The patterns are grouped in 13 random sets of 16 patterns each. According to cross validation scheme, 12 of these sets are used for training and the remaining set for testing, and this is repeated for 13 times changing every time the test set. In this way Gorman and Sejnowski trained a three-layer neural network with 12 hidden nodes. We cannot compete with the accuracy of the neural network, by definition. However we reduce to $1/5$ the length of the data representation paying less than 10 percentage points. Actually the classification problem is difficult *per se* as it is denoted by the length of the DNFs discovered by our method, on one side, and the high number of hidden nodes on the other side. Also the goal to hit is not univocal. As shown in Table 2 we may compress the data either into 50 Boolean variables, getting an optimal accuracy or into 60 Boolean variables, getting an optimal formula concision. Note that C4.5 is able to compute a shorter formula using the much longer original variables, but it pays in accuracy.

BICA Sonar				
Length			Correct %	
Signal	Formula			
	μ	σ	μ	σ
50	101.2	9.8	75.3	19.9
60	83.3	2.3	73.3	15.4

Table 2. Trade-off between different compression targets. Same notation as in Table 1.

3.3 Ionosphere database

The benchmark consists of 351 patterns of 34 real variables each. The (sole) training set consists of 100 patterns belonging to the class "good" and 100 to the class "bad", while remaining points (26 "good" and 125 "bad") are put in the test set. In this case we obtain almost the same formulas description length with our and C4.5 method, but we work with much compressed data. Actually it is the worst benchmark for us, but it may be biased by the particular training set selection.

3.4 The Monk's Problem

It is a set of three problems to explore the principal weakness points of a learning algorithm. The research space is made up of 6 features $(a1, \ldots, a6)$ ranging in discrete sets, two of them of cardinality 2, three of cardinality 3, and last

Alg. \ # vars.	7	8	9	10	17
Monk1	64.24	68.65	68.14	67.52	71.42
Monk2		66.95	68.47	73.94	76.32
Monk3		72.22	78.95	69.05	77.93

Table 3. Course of mean accuracy in percentage with length of Boolean assignment vectors; # vars: number of variables the features map onto.

one of cardinality 4 for a total of 432 records; the concepts to be discovered by analyzing *ad hoc* built training and testing sets are:

- Monk1: $(a1 = a2)$ or $(a5 = 1)$
- Monk2: EXACTLY TWO of $\{a1 = 1, a2 = 1, a3 = 1, a4 = 1, a5 = 1, a6 = 1\}$
- Monk3: $(a5 = 3$ and $a4 = 1)$ or $(a5 \neq 4$ and $a2 \neq 3)$ (5% class noise added to the training set)

The reference accuracies are obtained in [6] using ID3 algorithm [15] that is a previous version of C4.5. We add companion formula lengths (missing in the original paper) by rerunning the data on C4.5.

From Table 1 we see that the comparative classification accuracy, hence the capability of extracting relevant information from the data, grows with the complexity of the classification problem (M2 > M3 > M1, as denoted both by the above formalization of the problems, and by the length of the formulas discovered by C4.5). Namely, our procedure outperforms ID3 in M2 though requiring a very long formulas to manage the variables, while with simpler problems we pay. Table 3 shows that the drawback is almost monotone with the compression rate, specially with the most complex problem. HC gets generally the best accuracies with questionable effects on the lengths.

4. Conclusions

Meeting at a pub for the first time, a terrestrial and a Martian human (if such exists) may decide either to tell each other the beauty of their respective lives or to share a beer. We opt for the second task because it is driven by a common intention that we expect will supply a common codebook for interpreting the signals that they exchange [17, 13]. In this framework there is no discontinuity between data received and rules expressly built on so as to cope with a given task. However, we manage the process in two phases, symbol extraction and rule construction. We then focus on the former. To portray the efficiency of our compression method we also instantiated the second phase in a very elementary way, within the general philosophy that no data compression method is wonderful *per se* but depends rather on how suitably the compressed data may be used. In comparison with well assessed classification methods we discover that the compression of the data we propose has in general the side benefit of working with small formulas for classifying them, with a general bearable as expectable reduction of the classification accuracy.

References

[1] B. Apolloni, A. Esposito, D. Malchiodi, C. Orovas, G. Palmas, and J. Taylor. A general framework for learning rules from data. *IEEE Trans. on Neural Networks*, 15(6), 2004.

[2] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[3] C. M. Bishop. *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995.

[4] CORDIS. Ist thematic priority work programme, Accessed June 2005. http://www.cordis.lu/ist/workprogramme/fp6_workprogramme.htm.

[5] T. Cover and J. Thomas. *Elements of information theory*. Wiley, New York, 1991.

[6] S. T. et al. The monk's problems. Technical Report CMU-CS-91-1897, Carnegie Mellon University, Dec. 1991.

[7] A. Gersho and R. M. Gray. *Vector Quantization and Signal Processing*. Kluwer, Boston, MA, 1992.

[8] N. Guarino. Formal ontology and information systems. In *Formal Ontology in Information Systems*, pages 3–15, 1988.

[9] V. Hanover. Intelligent agents and multi-agent system. In *IEEE CEC*, 1999.

[10] N. Karayiannis. Reformulating learning vector quantization and radial basis neural networks. *Fundamenta informaticae*, 37:197, 1999.

[11] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag, Berlin, 2nd edition, 1988.

[12] M. Muselli and D. Liberati. Binary rule generation via Hamming clustering. *IEEE Trans. on Knowledge and Data Eng.*, 14:1258–1268, 2002.

[13] M. Nadin. What is the difference between a falling stone and a falling cat? *UBIQUITY*, 5, 2005.

[14] J. Pollack. Recursive distributed representation. *Artificial Intelligence*, 46:77–105, 1990.

[15] J. R. Qunilan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[16] V. K. Rohatgi. *An Introduction to Probability Theory and Mathematical Statistics*. John Wiley & Sons, NY, 1976.

[17] A. Rosen. *Anticipatory Systems*. Pergamon, 1985.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press, Cambridge, 1987.

[19] J. Zhang. Selecting typical instances in instance-based learning. In *Proc. of the ninth international workshop on Machine learning*, pages 470–479. Morgan Kaufmann, 1992.