THE PRINCIPAL AXES METHOD FOR CONSTRUCTIVE INDUCTION

by

J. Bala
R. S. Michalski
J. Wnek

In Proceedings of the 9th International Conference on Machine Learning, D. Sleeman and P. Edwards (Eds.), Aberdeen, Scotland, July 1992.

The Ninth International Conference on Machine Learning, Aberdeen, Scotland, July 1992.

The Principal Axes Method for Constructive Induction

Jerzy W. Bala, Ryszard S. Michalski and Janusz Wnek
Center for Artificial Intelligence, Department of Computer Science
George Mason University, Fairfax, VA 22030
{bala, michalski, wnek}@aic.gmu.edu

Abstract

The paper describes a novel method for constructive induction, called PRAX (Principal Axes). The underlying idea of the method is to determine descriptions of a class of certain basic concepts, and then use these descriptions as "principal axes" with which all other concepts can be described. Given examples of a new concept, the system determines a *similarity matrix* (SM) for that concept, that contains the average degree of similarity between the concept examples and the principal axes. These degrees of similarity are viewed as newly constructed attributes. To recognize an unknown concept instance, the method creates an SM for it, and then seeks the best matching similarity matrix of all known concepts. In experimental testing of the method on the problem of learning descriptions of a large number of visual textures, the PRAX method significantly outperformed the k-NN classifier often used for such problems. A very important result of this research is a demonstration that a symbolic learning method can be successfully applied to the domain of continuous attributes of low level vision in which nonsymbolic methods have been traditionally employed.

1 INTRODUCTION

The ability to inductively derive general patterns and laws from data is fundamental to human cognition and intelligence. Such ability is crucial for object recognition, classification, discovery of scientific laws and creativity. Most inductive learning methods developed so far create general descriptions that use attributes (generally, *descriptors*) that are selected from among those present in the original examples. Among known systems for such *selective* induction systems are AQVAL/1 (Michalski, 1973), ID3 (Quinlan, 1983), ASSISTANT (Cestnik, Kononenko & Bratko, 1987), and CN2 (Clark and Niblett, 1989). An advance over selective induction systems are *constructive induction* systems that are able to generate and use new descriptors in the hypothesized description. These new descriptors can be attributes, predicates, terms, functions, transformations, etc., which should be more relevant to the learning problem than those initially given. A constructive learning algorithm thus performs a problem-oriented transformation of the knowledge representation space (Michalski, 1978; Rendell, 1985; Matheus, 1989; Drastal, Czako & Raatz, 1989).

The idea of constructive induction was first proposed by Michalski (1978), and implemented in the INDUCE.1 system for learning structural descriptions from examples. INDUCE.1 used a variety of constructive *generalization rules* to generate new descriptors. These rules included a "replacing a property by an implied property" rule, various "counting arguments" rules, the "generating chain properties" rule, and "detecting a descriptor interdependence" rules (see also Michalski, 1983). Subsequently, a number of other systems that exhibit constructive induction capabilities have been developed (Bloedorn & Michalski, 1991; Wnek & Michalski, 1991).

Current constructive induction methods can be divided into four categories:

- **Data-Driven (DCI)** - by analyzing and exploring the input data.
  Example systems:
  INDUCE (Michalski, 1978)
  LEX (Mitchell, Utgoff & Banerji, 1983)
  AM, EURISCO (Lenat, 1983)
  BACON (Langley, Bradshaw & Simon, 1983)
  STAGGER (Schlimmer, 1987)
  AQ17-DCI (Bloedorn & Michalski, 1991)

- **Hypothesis-Driven (HCI)** - by analyzing generated hypotheses.
  Example systems:
  FRINGE (Pagallo & Haussler, 1990) - for decision trees
  AQ17-HCI (Wnek & Michalski, 1991) - for decision rules

- **Knowledge-Driven (KCI)** - by applying expert-provided knowledge.
  Example systems:
  INDUCE (Michalski, 1978)
  AQ15 (Michalski, Mozetic, Hong & Lavrac, 1986)
  MIRO (Drastal, Czako & Raatz, 1989)
  DUCE, CIGOL (Muggleton, 1987)

- **Multistrategy (MCI)** - by integrating several methods.
  Example systems:
  PLS0 (DCI, KCI; Rendell, 1985)
  CITRE (DCI & KCI; Matheus, 1989)
  AQ17 (DCI, HCI, KCI--A-rules & L-rules, GDN*; (Stepp & Michalski, 1986); Bloedorn, Michalski & Wnek, 1992).

In an attribute-based learning system, the abstraction level of attributes strongly affects the complexity of the hypothesized rules. By employing high-level attributes, the concept representation can be simplified. Such attributes may, however, be encoded as very complex functions of low-level primitives. In such cases, to improve the efficiency, these complex functions have to be compiled (Flann & Dietterich 1986). Constructive induction may lead to the maximal simplification of the generated descriptions and/or improvement in the predictive accuracy of the hypothesis. The predictive accuracy can be measured by applying the hypothesis to the testing data, and determining the correctness of the predictions.

The PRAX method represents a form of the hypothesis-driven constructive induction (Wnek & Michalski, 1991). Constructive induction of this type analyzes inductive hypothesis in order to transform an initial descriptor set into a new, problem-relevant descriptor set. In general, the transformation of a descriptor set may involve generalization or specialization. One may generalize the set by dropping (removing) attributes or narrowing their domains, or one may specialize the set, by adding new attributes or extending domains of existing attributes. More specifically, construction of a new descriptor may involve detecting four types of attribute patterns: null-pattern, condition-pattern, rule-pattern, and ruleset-pattern. *Null-pattern* of an attribute is detected when the attribute is used in few or no conditions of the hypothesis. In this case, the attribute can be removed from the descriptor set without loosing the ability to properly express target concept. *Condition-patterns* are detected, if the analyzed hypothesis contains conditions that repeatedly involve the same values of a single attribute. *Rule-patterns* involve conjunctions of conditions. *Ruleset-patterns* are found in sets of rules, i.e., disjunctions of rules. The new attributes can have either binary or real values.

The PRAX method uses rule-pattern attributes with a real valued similarity measure. This learning process is accomplished in two phases that represent two major transformations on the representation of the domain. In the first phase the system learns discriminant descriptions of initial concepts. These descriptions are called principal axes. In the second phase the principal axes are used to incrementally learn new concepts. Any newly acquired description of a concept is represented as a similarity measure between already learned Principal Axes description and instances of the new concept. These similarity measures are represented as a matrix data structure, called Similarity Matrix (SM).

An empirical evaluation of the method is presented using the texture data (24 texture classes represented in the eight dimensional feature space). The recognition results of the PRAX method are compared with the results obtained from the k-NN classifier. The next three sections describe two phases of the method; learning Principal Axes and learning Similarity Matrix. Section 5 explains the constructive induction aspect of the presented method and the empirical evaluation is presented in section 6.

2 LEARNING PRINCIPAL AXES

The problem-oriented descriptors - principal axes - are learned using the standard AQ algorithm (Michalski, 1973). Specifically, the algorithm is used to generate a discriminant description of initial concepts (Principal Axes). The concept descriptions learned by AQ algorithm are represented in VL_1, which is a simplified version of the Variable-Valued Logic System VL, and are used to represent attributional concept descriptions. In the application of machine learning to texture recognition described in this paper, a concept represents a single texture class. A description of a concept is a disjunctive normal form which is called a cover. Below is an example of a cover generated by the AQ module:

[x1=10..54] & [x3=18..54] & [x5=11..17] & [x6=6] *or*
[x3=18..54] & [x4=16..54] & [x6=0..6] & [x8=5..12]

* GDN stands for *Goal Dependency Network* that was originally introduced in (Stepp & Michalski, 1986) for generating problem-relevant attributes for conceptual clustering.

The above cover (disjunctive normal form) consists of two disjuncts. It covers learning instances represented by eight attributes {x1 .. x8}. For example, the learning instance <20, 10, 25, 17, 1, 4, 30, 6> is covered by the second disjunct. The attributes values x3=25, x4=17, x6=4, x8=6 of that instance are covered by conditional parts of that disjunct and all other attributes {x1, x2, x5, x7} are also covered because conditional parts of these attributes are not present in the disjunct.

The AQ cover is optimized by using the truncation method that was first introduced by Michalski et al. (1986). In the AQ cover, each generated disjunct is associated with a pair of weights; i.e., the t-weight (as the total number of positive examples covered by the disjunct) and the u-weight (as the number of positive examples uniquely covered by the disjunct). The disjuncts are ordered according to the decreasing values of the t-weight. The t-weight may be interpreted as the measure of typicality or the representativeness of a disjunct as a concept description. The disjuncts with the highest t-weight may be viewed as describing the most typical concept examples, and thus serve as its prototypic description. The disjuncts with the lowest t-weight are truncated from the description.

3 DISTANCE METRIC

The method uses a non-linear distance metric to calculate values of new attributes. The distance metric is based on the idea of flexible matching. In flexible matching, the degree of closeness between the example and the concept is determined, instead of a binary decision as used in strict matching. Specifically, the match of an example E to a disjunct D is computed by the following formula:

$$\text{MATCH}(E, D) = \prod_i \left(1 - \frac{\text{dis}(E_i, D_i)}{\text{max}_i - \text{min}_i}\right)$$

where E_i is the value of the i-th attribute of example E, D_i is the condition involving the i-th attribute in D, max_i and min_i are the maximum and minimum values of the ith attribute, and m is the number of attributes. $\text{dis}(E_i, D_i)$ depends on the type of the attribute involved in the condition. An attribute can be one of two types: nominal and linear. In a nominal condition, the referent in a condition is a single value or an internal disjunction of values, e.g., [color = red v blue v green]. The distance is 1, if such a condition is satisfied by an example, and 0 if it is not satisfied. In a linear condition, the referent is a range of values, or an internal disjunction of ranges, e.g., [weight = 1..3 v 6..9]. A satisfied condition returns the value of distance 1. If the condition is not satisfied, the distance between an example and the condition is the absolute of a difference between the value of the example and the nearest end-point of the interval of the condition normalized by the distance between the farthest value and the condition. For example, if the domain of x is [0 .. 10], the value of x for the example E is E_x=4 and the condition is [x = 7 .. 9], then

$$\text{dis}(E_x, \text{condition}) = \frac{7 - 4}{10 - 0} = \frac{3}{10}$$

A flexible matching method is used to calculate the degree of match. To illustrate the method the following example is presented below.

Given:

disjunct D	[x1=1..3] & [x2=1] & [x4=0] & [x6=1..7] & [x8=1]
example E	<2, 4, 2, 1, 6, 10, 7, 5,>
# of attributes:	m = 8
attribute range	$\text{max}_i - \text{min}_i$ = 55

The degree of flexible match to the example is calculated by evaluating partial matches of a given attribute value of the example to the corresponding conditional part of the disjunct. Evaluations for each attribute are:

dis_{x1}	=	1	(an attribute value is covered)
dis_{x2}	=	1 -	| 1 - 4 | / 55 = 0.945
dis_{x3}	=	1	(x3 not present in the disjunct)
dis_{x4}	=	1 -	| 0 - 1 | / 55 = 0.981
dis_{x5}	=	1	(same as for x3)
dis_{x6}	=	1 -	| 7 - 10 | / 55 = 0.945
dis_{x7}	=	1	(same as for x3)
dis_{x8}	=	1 -	| 5 - 1 | / 55 = 0.927

and, the degree of match to the disjunct is

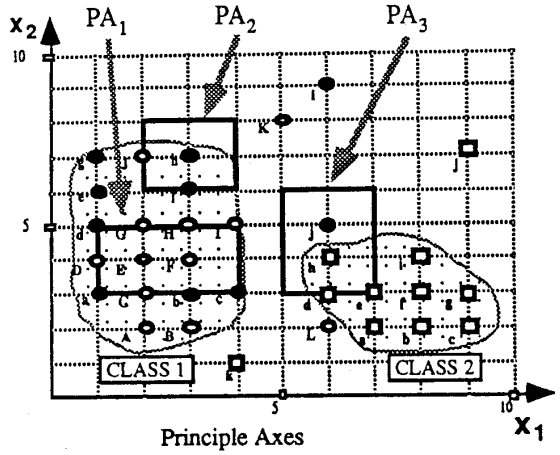
$$\text{MATCH}(E, D) = \prod_i \text{dis}_{xi} = 0.812$$

The flexible match method as described above is used in generating the Similarity Matrix (SM) description, i.e. the concept description expressed in the new representation space. The SM description is obtained by applying the flexible matching process to the examples of the new concept and the previously learned Principal Axes.

4 GENERATING SIMILARITY MATRIX

The mechanism of determining SM description is illustrated in Figure 1 by a simple example in a two dimensional feature space. Black dots are used to indicate the learning examples of CLASS 1 and white dots are the testing examples of CLASS 1. White rectangles are

learning examples of CLASS 2 . Generation of SM is based on the calculation of the degree of match between instances of CLASS 1 and CLASS 2 to three disjuncts: PA_1, PA_2, and PA_3. These disjuncts represent the Principal Axes learned in the previous step by using examples of some initial classes (examples of that classes are not depicted in the Figure 1). The process of generating SMs is illustrated in Table 1. Table 1 shows also how the generated SM descriptions are used to recognize CLASS 1 testing examples (white dots). The following are two SMs generated from learning examples. They represent an average (normalized to 100) match to three principal axes (PA_1, PA_2, and PA_3)



Principle Axes

PA_1: [x1=1..4] & [x2=3..5]

PA_2: [x1=2..4] & [x2=6..8]

PA_3: [x1=5..7] & [x2=3..6]

Figure 1: Two Classes Example.

CLASS 1 SM = [86, 80, 73]
CLASS 2 SM = [61, 47, 83]
SM generated from testing examples of CLASS 1 is
SM-test = [91, 80, 74]
The following is the calculation to determine class membership of testing examples:

Match(Test , CLASS 1) =
Match([91, 80, 74] , [86, 80, 73]) =
(abs(91-86) + abs(80-80)+abs(74-73))/3 =
(5+0+1)/3=2

Match(Test , CLASS 2) =
Match([91, 80, 74] , [61, 47, 83]) =
(abs(91-61)+abs(80-47)+abs(74-83))/3 =
(30+33+9)/3=24
Testing examples are matched by CLASS 1.

5 DESCRIPTION SPACE TRANSFORMATION

The process of learning texture descriptions, in the PRAX method, imposes two major transformations on the representation of the domain. The starting, intermediate, and final descriptions are expressed within three level language. Higher levels extend properties of lower levels by grouping features from the previous level. The domain knowledge is primarily expressed in terms of eight features. These features represent eight Laws masks*. Each of them already carries meaningful information, relevant to the texture domain. However, when considered separately, these features do not sufficiently discriminate among many classes of textures. (It would be the same as trying to distinguish texture classes by using only one property, e.g. density of horizontal edges).

Thus, the second level language facilitates building conjunctive expressions of feature values used in the first place. This allows for the expressing of discriminatory features of textures with respect to relevant values of all features, but still lacks sufficient precision in describing highly flexible, imprecise concepts, such as textures. Therefore, the expressions found, are not used as final descriptions of the class concepts, but they serve as dynamically constructed features for building final descriptions. The features, found in this step, structure the knowledge contained in the training examples of different texture classes. Feature descriptions reflect important relations between selected feature values in accordance with the goal of being discriminable among given classes. At this point, the problem of finding texture classes descriptions receives proper orientation and is encoded into the new features, which are relevant to the learning goal. The new features are assumed to have continuous values within the 0 to 1 range (a degree of match to a disjunct, in the SM description they are normalized in the range 0..100).

The final concept description is expressed in the form of the Similarity Matrix. The size of the matrix - the

* The most frequently used methods of texture feature extraction are based on the analysis of a local subset of pixels. This subset is defined by a small window in order to derive local characteristics of covered pixels. There are many methods applied to derive local characteristics of pixels. Since the development of low-level image processing techniques is not in scope of this research, we decided to apply well known and well performing low-level operators to extract texture features. Laws' energy filters (Laws, 1980) were chosen to transform raw image data into texture features. Laws masks are the most often used features in the texture analysis.

The first column represents examples as shown in Figure 1 (e.g. a, b, c, ...).
Other columns represent the degree of match of these examples to disjuncts (axes) PA_1, PA_2, PA_3.
SMs are normalized to the 1..100 range of average degrees of match to each disjunct.

CLASS 1 learning examples (black dots in Figure 1)

Example (E)	MATCH (E, PA_1)	MATCH (E, PA_2)	MATCH (E, PA_3)
a	1	(1-1/10)*(1-3/10)=0.63	(1-4/10)*1=0.6
b	1	1*(1-3/10)=)=0.7	(1-2/10)*1=0.8
c	1	1*(1-3/10)=)=0.7	(1-1/10)*1=0.9
d	1	(1-1/10)*(1-1/10)=0.81	(1-4/10)*1=0.6
e	1*(1-1/10)=0.9	(1-1/10)*1=0.9	(1-4/10)*1=0.6
f	1*(1-1/10)=0.9	1	(1-2/10)*1=0.8
g	1*(1-2/10)=0.8	1-1/10)*1=0.9	(1-4/10)*(1-1/10)=0.54
h	1*(1-2/10)=0.8	1	(1-2/10)*(1-1/10)=.72
i	(1-2/10)*(1-4/10)=0.48	(1-2/10)*(1-1/10)=0.72	1*(1-3/10)=0.7
j	(1-2/10)*1=0.8	(1-2/10)*(1-1/10)=0.72	1
Average	0.86	0.80	0.73

CLASS 1 SM = [86, 80, 73]

CLASS 2 learning examples (white rectangles in Figure 1)

Example (E)	MATCH (E, PA_1)	MATCH (E, PA_2)	MATCH (E, PA_3)
a	(1-3/10)*(1-1/10)=0.63	(1-3/10)*(1-4/10)=0.42	1*(1-1/10)=0.9
b	(1-4/10)*(1-1/10)=0.54	(1-4/10)*(1-4/10)=0.36	(1-1/10)*(1-1/10)=0.81
c	(1-5/10)*(1-2/10)=0.4	(1-5/10)*1=0.5	(1-2/10)*(1-1/10)=0.45
d	(1-2/10)*1=0.8	(1-2/10)*(1-3/10)=0.56	1
e	(1-3/10)*1=0.7	(1-3/10)*(1-3/10)=0.49	1
f	(1-4/10)*1=0.6	(1-4/10)*(1-3/10)=0.42	(1-1/10)*1=0.9
g	(1-5/10)*1=0.5	(1-5/10)*(1-3/10)=0.35	(1-2/10)*1=0.8
h	(1-2/10)*1=0.8	(1-2/10)*(1-2/10)=0.64	1
i	(1-4/10)*1=0.6	(1-4/10)*(1-2/10)=0.48	(1-1/10)*1=0.9
j	(1-5/10)*(1-2/10)=0.4	(1-5/10)*1=0.5	(1-2/10)*(1-1/10)=0.72
k	1*(1-2/10)=0.8	1*(1-5/10)-0.5	(1-1/10)*(1-2/10)=0.72
Average	0.61	0.47	0.83

CLASS 2 SM = [61, 47, 83]

The following is the calculation of the test SM description
CLASS 1 test examples (white dots in Figure 1)

Example (E)	MATCH (E, PA_1)	MATCH (E, PA_2)	MATCH (E, PA_3)
A	1*(1-1/10)=0.9	1*(1-4/10)=0.6	(1-3/10)*(1-1/10)=0.63
B	1*(1-1/10)=0.9	1*(1-4/10)=0.6	(1-2/10)*(1-1/10)=0.72
C	1	1*(1-3/10)=0.7	(1-3/10)*1=0.7
D	1	1*(1-2/10)=0.8	(1-4/10)*1=0.6
E	1	1*(1-2/10)=0.8	(1-3/10)*1=0.7
F	1	1*(1-2/10)=0.8	(1-2/10)*1=0.8
G	1	1*(1-1/10)=0.9	(1-3/10)*1=0.7
H	1	1*(1-1/10)=0.9	(1-2/10)*1=0.8
I	1	1*(1-1/10)=0.9	(1-1/10)*1=0.9
J	1*(1-2/10)=0.8	1	(1-3/10)*(1-1/10)=0.63
K	(1-1/10)*(1-3/10)=0.63	(1-1/10)*1=0.9	1*(1-2/10)=0.8
L	(1-2/10)*(1-1/10)=0.72	(1-2/10)*(1-1/10)=0.72	1*(1-1/10)=0.9
Average	0.91	0.80	0.74

Test SM = [91, 80, 74]

Table 1: Illustration of Learning And Recognition For the Two Classes Case From Figure 1.

number of columns - depends on the number of the initial training classes. The number of rows depends on the number and generality of useful conjunctive features describing class represented in a column.

The process described above can be viewed as a two step induction: (1) learning a new, problem-relevant descriptor set from examples of the initial texture classes, and (2) learning a description of a given class using descriptors from the previous step and examples of that class (Figure 2).

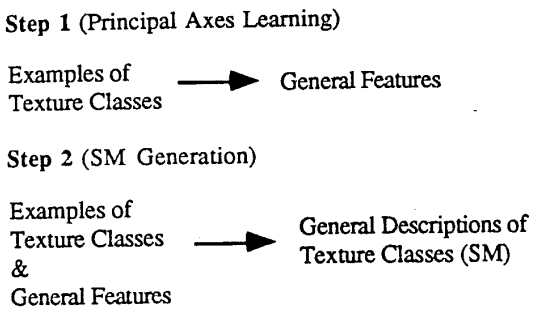


Figure 2: Two-step Inductive Learning

In the example presented in Figure 1 and Table 1 the initial features {x1, x2} are transformed into new features $\{PA_1, PA_2, PA_3\}$. This transformation can be expressed as follows:

$T(\{x1, x2\}) \Longrightarrow$

$\{ MATCH(\{x1,x2\}, PA_1),$
$MATCH(\{x1,x2\}, PA_2),$
$MATCH(\{x1,x2\}, PA_3) \}$

Corresponding feature values in two domains for three examples of CLASS 1 (a, b, and c black dots in Figure 1) are shown below

Initial example		New example
a: {1 , 3}	-->	{1, 0.63, 0.6}
b: {3 , 3}	-->	{1, 0.7, 0.8}
c: {4 , 3}	-->	{1, 0.7, 0.9}

In the experiment with texture classes the initial eight features (eight Law's masks) {x1,..,x8} are transformed into 170 features $\{ PA_1,..,PA_{170} \}$ that represent 170 disjuncts (Principal axes) learned by the AQ method using instances of eight texture classes.

6 EMPIRICAL EVALUATION

In experiments we used 24 texture classes. A texture class was encoded as an image file 512 by 512 pixels. From each class two sets of examples were extracted. The first set, with 200 examples, was used as the learning data and the second set, with 200 examples, was used as the testing data. Each examples of a given texture was represented using 8 features. Feature values were extracted using Laws masks. All feature values were scaled to the linear range 0 to 54 (this is the range of linear attributes that the AQ algorithm can handle). The learning data was extracted from the left hand side of the texture image and the testing data from the right hand side. For each run of the experiment different learning and testing data sets were used. Eight classes (C1 to C8 in Figure 3) were chosen to learn the Principal Axes. The learned Principal Axes after truncation consisted of 115 disjuncts. Examples of all 24 classes (including initial eight) were used to generate SM description. All SM descriptions were stored as the learned descriptions of texture classes. In the recognition phase 200 testing examples of each class (extracted from a different texture area than the one used for the learning examples) were used to generate an SM description to test the recognition accuracy. SM descriptions obtained from the testing examples were matched (a simple matching calculations are presented in section 4) with the SM descriptions obtained from the learning examples. Results of a one run of the experiment are presented in Table 2. An example of SM descriptions for class C9 obtained from learning and testing examples are presented in Appendix.

The method was compared with the k-NN (k-Nearest Neighbor) classifier. During five experiments (in each experiments we used different learning and testing sets) we obtained on average one class mis-classified (for k=1 to 10). For the next step the two methods with mis-classification noise present were compared. A 10% mis-classification noise was introduced to the texture data (10 % of learning examples of a given class were shuffled from other classes). The k-NN classifier was unable to correctly recognize two classes (C1 and C3) while the PRAX method correctly recognized all 24 classes. The first degradation of the method was observed with 20% of mis-classification noise (on average 2 classes not recognized by the PRAX method and 7 classes not recognized by k-NN classifier).

7 SUMMARY

The presented method consists of two phases that represent two major transformations on the representation of the domain. The initial phase of the method determines a set of discriminant descriptions of a class of basic

Tested Class	Closest Match	Degree	Second Closest	Degree	Worse Match	Degree
C1	C1	2.36	C21	11.28	C8	47.64
C2	C2	11.4	C22	14.59	C8	49.21
C3	C3	5.15	C13	12.97	C14	51.55
C4	C4	7.70	C9	15.65	C15	38.31
C5	C5	5.17	C6	18.87	C8	40.04
C6	C6	2.62	C5	19.89	C24	45.60
C7	C7	8.21	C13	9.52	C14	44.63
C8	C8	4.60	C10	6.30	C22	51.20
C9	C9	2.60	C10	13.24	C15	52.53
C10	C10	0.96	C8	3.16	C22	50.99
C11	C11	9.41	C22	12.59	C8	43.92
C12	C12	4.41	C24	5.55	C15	43.69
C13	C13	4.62	C18	7.01	C8	47.94
C14	C14	3.86	C18	8.83	C9	47.77
C15	C15	5.31	C3	11.78	C14	43.07
C16	C16	7.69	C8	5.60	C22	51.06
C17	C17	4.46	C6	13.08	C13	50.71
C18	C18	2.59	C7	14.09	C14	46.95
C19	C19	3.31	C5	21.34	C3	55.99
C20	C20	5.74	C18	7.23	C10	52.18
C21	C21	4.73	C23	8.83	C14	44.87
C22	C22	3.35	C22	8.42	C10	48.87
C23	C23	2.02	C36	7.36	C8	50.19
C24	C24	1.99	C11	4.18	C12	46.65

Table 2: Recognition results of one run of the experiment. The third column represents the value of the closest degree of match. This value is calculated as an average difference between corresponding entries of the learned and test SM descriptions. An example of the such calculation is depicted in Table 1 for a simple two feature domain. The second closest and the worse matches are also included in the table below to show the range of degree of match values for all classes.

concepts selected by a teacher. The descriptions of these concepts constitute the principal axes that serve as constructed attributes for expressing any new concept descriptions. A description of any new concept is represented as a similarity matrix that specifies the degrees of similarity between the examples of the new concept and the principal axes. Experimental results of texture recognition were presented to illustrate the feasibility of our novel methodology. The results are very encouraging when compared with results presented in other computer vision papers on texture recognition (they do not report on recognition of more than 20 textures).

The main strength of the method lies in a problem-relevant transformation of the descriptor space. The new descriptors form generalized sub-spaces of the initial, training space. In addition, the method uses a non-linear distance metric to calculate values of new attributes. The distance metric based on the idea of flexible matching is less sensitive to noise, then traditional Euclidean distances used by pattern recognition methods.

In experimental testing of the method on the problem of learning descriptions of a large number of visual textures, the PRAX method outperformed the k-NN classifier.

The current problem with the method is that it does not have the mechanism to decide how to choose basic concepts. Choosing the minimal subset of concepts to be used for principal axes generation is crucial for method optimization. This problem is subject to further research.

Acknowledgments

We would like to express our gratitude to Eric Bloedorn for the discussions and comments about this work. This research was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by the Defense Advanced Research Projects Agency under the grants administered by the Office of Naval Research No. N00014-87-K-0874 and No. N00014-91-J-1854, in part by the Office of Naval Research under grants No. N00014-88-K-0397, No. N00014-88-K-0226 and No. N00014-91-J-1351, and in part by the National Science Foundation Grant No. IRI-9020266.

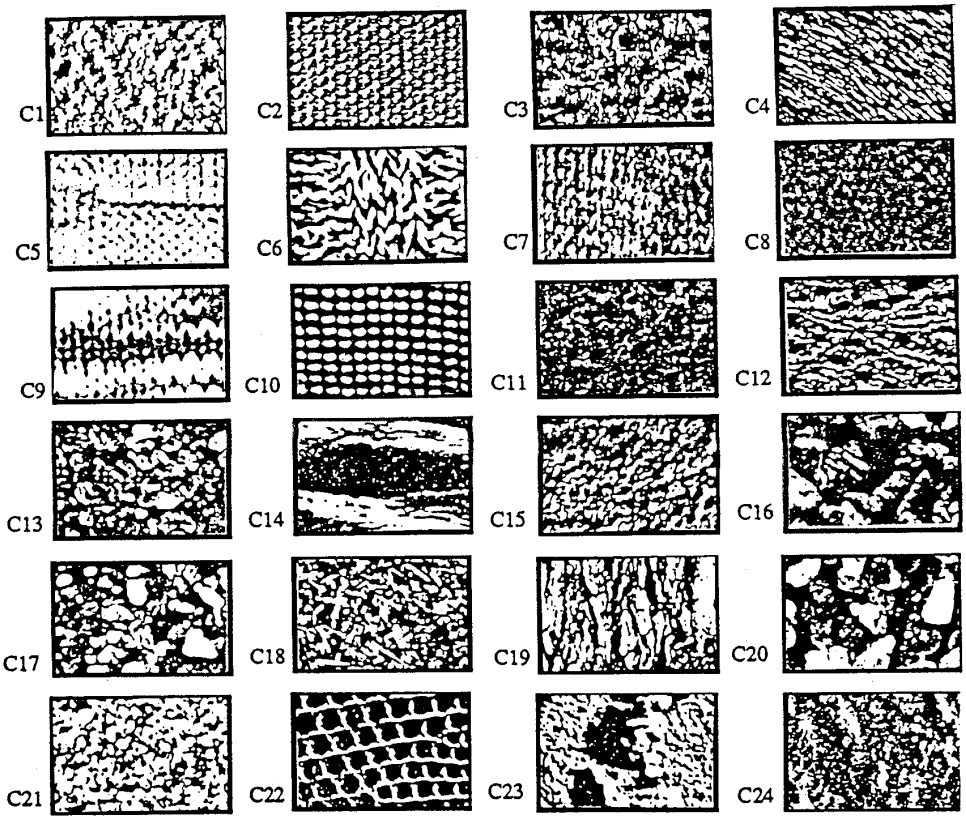


Figure 3: Samples From the Learning Area of 24 Textures.
Approximately 40 x 40 Pixels (Brodatz Album of Textures, 1966).

References

Bloedorn, E. and Michalski, R.S., "Data-Driven Constructive Induction in Method and Experiments," *Proceedings of the 3rd International Conference on Tools for AI*, San Jose, CA, pp.30-37, 1991.

Bloedorn, E., Michalski, R.S. and Wnek, J., "AQ17 - A Multistrategy Constructive Learning System," to appear in *Reports of Machine Learning and Inference Laboratory*, Center for Artificial Intelligence, George Mason University, 1992.

Brodatz, P., Textures: A Photographic Album for Artists and Designers, New York, 1966.

Cestnik, B., Kononenko, I. and Bratko, I., "ASSISTANT 86: A Knowledge Elicitation Tool for Sophisticated Users," *Proceedings of EWSL-87*, Bled, Yugoslavia, pp. 31-45, 1987.

Clark, P., and Niblett, T., "The CN2 Induction Algorithm," *Machine Learning*, **3**, pp. 261-284, 1989.

Drastal, G., Czako, G. and Raatz, S., "Induction in an Abstraction Space: A Form of Constructive Induction," *Proceedings of the IJCAI-89*, pp. 708-712, Detroit, MI, August 1989.

Flann, N.S., and Dietterich, T.G., "Selecting Appropriate Representations for Learning from Examples," *Proceedings of AAAI-86*, Philadelphia, PA, pp. 460-466, 1986.

Langley, P, Bradshaw, G.L. and Simon, H.A., "Rediscovering Chemistry With the BACON System," *Machine Learning: An Artificial Intelligence Approach*, Vol. I, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (eds.), Morgan Kaufmann, 1983.

Laws, K.I., "Textured Image Segmentation", Ph.D. Thesis, Dept. of Electrical Engineering, University of Southern California, Los Angeles, 1980.

Lenat, D.B., "Learning from Observation and Discovery," *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, Vol. I, J.G. Carbonell, and T.M. Mitchell (eds.), Morgan Kaufmann, 1983.

Matheus, C., "Feature Construction: An Analytic Framework and Application to Decision Trees," *Ph.D. Thesis*, University of Illinois, 1989.

Michalski, R.S., "AQVAL/1 - Computer Implementation of a Variable-Valued Logic System VL1 and Examples of its Application to Pattern Recognition," *Proceedings of the First International Joint Conference on Pattern Recognition*, Washington, D.C., pp. 3-17, 1973.

Michalski, R.S., "Pattern Recognition as Knowledge-Guided Computer Induction," Report No. 927, Department of Computer Science, University of Illinois, Urbana, June 1978. (Published under the title "Pattern Recognition as Rule-Guided Inductive Inference," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, pp. 349-361, July 1980.)

Michalski, R. S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, Vol.20, pp.111-161, 1983.

Michalski, R.S., I. Mozetic, J. Hong and N. Lavrac, "The AQ15 Inductive Learning System: An Overview and Experiments", ISG 86-23, UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois, Urbana, 1986.

Mitchell, T.M., Utgoff, P.E. and Banerji, R., "Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics," in *Machine Learning: An Artificial Intelligence Approach*, Vol. I, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.), Morgan Kaufmann, 1983.

Muggleton, S., "Duce, and Oracle-Based Approach to Constructive Induction," *Proceedings of IJCAI-87*, pp.287-292, Milan, Italy, 1987.

Pagallo, G. and Haussler, D., "Boolean Feature Discovery in Empirical Learning," *Machine Learning*, 5, pp. 71-99, 1990.

Quinlan, J.R., "Learning Efficient Classification Procedures and Their Application to Chess End Games," *Machine Learning: An Artificial Intelligence Approach*, Vol. I, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (eds.), Morgan Kaufmann, 1983.

Rendell, L., "Substantial Constructive Induction Using Layered Information Compression:Tractable Feature Formation in Search," *Proceedings of IJCAI-85*, pp. 650-658, 1985.

Schlimmer, J.C, "Incremental Adjustment of Representations in Learning," *Proceedings of the Fourth International Machine Learning Workshop*, pp.79-90, 1987.

Stepp, R. and Michalski, R.S., "Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects," *Machine Learning: An Artificial Intelligence Approach*, Vol. II, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (eds.), Morgan Kaufmann, 1986.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," *Proceedings of the IJCAI-91 Workshop on Evaluating and Changing Representations in Machine Learning*, K.Morik, F.Bergadano and W.Buntine (Eds.), pp.13-22, Sydney, Australia, 1991.

Appendix

C9-learned SM

89.2	51.3	85.4	54.8	63.5	48.6	95.5	60.9
61.9	74.4	67.8	15.9	49.3	31.4	71.3	66.4
68.3	61.3	79.3	0.6	55.1	39.7	75.0	73.1
60.0	58.1	69.8	0.2	57.0	36.6	-	72.1
86.7	48.7	64.9	48.1	-	12.3	-	46.3
65.3	55.3	78.9	29.1	-	24.2	-	74.8
69.8	80.6	75.9	5.9	-	41.9	-	55.3
62.7	56.2	70.2	51.5	-	45.3	-	31.1
55.5	84.2	65.5	31.3	-	48.2	-	49.4
65.3	35.1	65.5	20.5	-	38.3	-	67.3
86.2	48.6	42.2	-	-	17.8	-	56.6
63.9	0.1	48.5	-	-	60.5	-	3.0
77.9	45.2	43.2	-	-	60.6	-	60.0
86.9	29.7	81.8	-	-	59.9	-	88.7
90.2	80.3	63.7	-	-	68.4	-	68.9
71.4	86.6	84.6	-	-	6.2	-	36.2
69.9	49.1	73.7	-	-	18.5	-	70.9
81.9	54.7	-	-	-	39.9	-	40.6
55.2	65.7	-	-	-	51.6	-	-
65.3	-	-	-	-	13.9	-	-
-	-	-	-	-	50.8	-	-
-	-	-	-	-	62.6	-	-
-	-	-	-	-	54.4	-	-
-	-	-	-	-	26.6	-	-

C9-test SM

88.3	46.5	88.5	55.0	69.5	44.4	95.4	60.7
61.2	71.4	70.2	15.9	56.1	27.0	69.7	68.3
69.5	57.0	83.1	3.8	59.5	39.7	73.2	73.2
60.2	52.8	71.4	0.2	61.5	35.0	-	71.9
87.0	45.7	67.6	48.4	-	10.0	-	47.1
63.4	52.2	81.7	28.0	-	22.5	-	74.4
70.8	78.6	78.3	4.8	-	38.8	-	58.0
61.6	50.3	72.2	49.7	-	42.1	-	26.6
60.0	83.3	70.1	31.2	-	45.2	-	45.1
63.5	30.8	69.2	16.4	-	33.8	-	66.4
87.6	46.6	45.8	-	-	16.8	-	58.8
64.1	0.1	50.9	-	-	61.2	-	54.6
74.8	40.5	44.3	-	-	59.4	-	59.3
88.4	27.8	83.1	-	-	59.4	-	88.4
91.7	79.4	66.4	-	-	67.5	-	69.5
70.0	87.1	87.5	-	-	4.3	-	32.9
71.4	45.9	76.5	-	-	13.2	-	67.0
82.4	51.4	-	-	-	39.3	-	40.4
56.4	65.6	-	-	-	49.4	-	-
66.9	-	-	-	-	12.1	-	-
-	-	-	-	-	49.1	-	-
-	-	-	-	-	60.4	-	-
-	-	-	-	-	49.8	-	-
-	-	-	-	-	22.2	-	-

Two SM descriptions for class C9 obtained from training and testing examples. Each entry in the SM description is an average degree of match (normalized to the range 0 to 100) of 100 learning examples of the class C9 to a given disjunct (170 disjuncts were learned in the first phase of the method from examples of classes C1 to C8 and 55 lightest disjuncts covering on average 20% of examples of each class were truncated leaving 115 disjuncts). The class C9 is the first new incrementally learned class that has only SM description. Classes C1 to C8 have both the rule-based (the AQ learned descriptions) and SM descriptions.