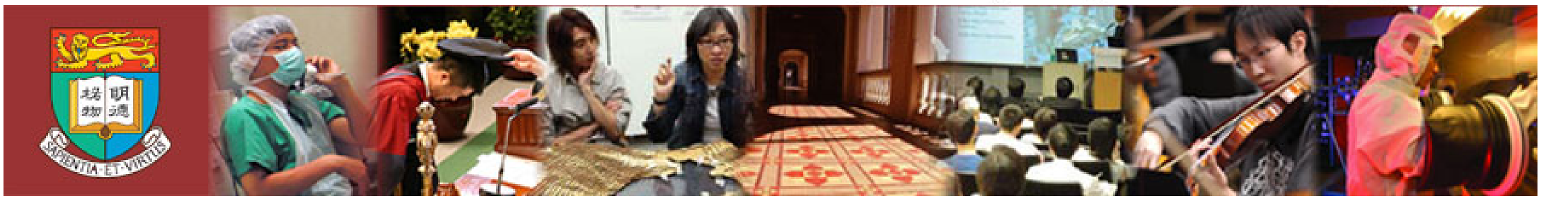
The HKU Scholars Hub The University of Hong Kong 香港大學學術庫



Title	Learning templates from fuzzy examples in structural pattern recognition
Author(s)	Chan, KwokPing
Citation	Ieee International Conference On Fuzzy Systems, 1994, v. 1, p. 608-613
Issued Date	1994
URL	http://hdl.handle.net/10722/43632
Rights	©1996 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Learning Templates from Fuzzy Examples in Structural Pattern Recognition

Kwok-Ping Chan, *Member IEEE*

***Abstract*—Fuzzy-Attribute Graph (*FAG*) was proposed to handle fuzziness in the pattern primitives in structural pattern recognition [1]. *FAG* has the advantage that we can combine several possible definition into a single template, and hence only one matching is required instead of one for each definition. Also, each vertex or edge of the graph can contain fuzzy attributes to model real-life situation. However, in our previous approach, we need a human expert to define the templates for the fuzzy graph matching. This is usually tedious, time-consuming and error-prone. In this paper, we propose a learning algorithm that will, from a number of fuzzy examples, each of them being a FAG, find the smallest template that can be matched to the given patterns with respect to the matching metric.**

I. Introduction

FUZZY set theory was first introduced by Prof. L. A. Zadeh in 1965 [2]. It is used as a formal mathematical tool to investigate problems pertaining to uncertainty, ambiguity and vagueness. The concepts that are modelled has no exact boundary between membership and nonmembership and the change between them is gradual rather than abrupt.

In classical set theory, an element either belongs to or does not belong to a set. This kind of precise knowledge provides us with rigor and completeness. However, in everyday life, we often do not want to deal with things precisely. For example, we would say "John is tall" rather than "John is 1.82 meters high" In fact, many features that are used for pattern recognition are fuzzy in nature, such as *tallness* of a man or *hotness* of an oven. Although these features can be expressed in precise measurement, human seldom use the exact values in decision making. Instead, abstraction is performed and these features are represented by fuzzy concepts, such as "tall", "short", "hot" and "cold" etc. They do not have an exact boundary between membership and nonmembership and the change is gradual. Hence fuzzy set theory provides us a rigorous mathematical tool to formally model these concepts. Moreover, an ordinary set can be represented as a special case of a fuzzy set. If $\tilde{A}$ is the fuzzy set that represents the ordinary set A, and $\mu_{\tilde{A}}(x)$ denotes the membership of x in the fuzzy set $\tilde{A}$, then

$$\mu_{\tilde{A}}(x) = \begin{cases} 1 & \forall x \in A \\ 0 & \text{otherwise.} \end{cases}$$

Manuscript received May 7, 1993; revised May 19, 1994, and December 10, 1994.

The author is with the Department of Computer Science, The University of Hong Kong, Hong Kong.

Publisher Item Identifier S 1083-4419(96)00413-X.

Although fuzzy set theory and probability both take values between 0 and 1, there are subtle difference between them. Also there has been argument that fuzzy set theory can be modelled by subjective probability. However, if one thinks carefully, the statement "John is tall with fuzzy membership 0.7" (fuzzy set theory) and "My belief that John is tall is 0.7" (subjective probability) is actually quite different. In the former case, the concept "tall" is fuzzy (gradual and no abrupt change in membership) while in the latter, 0.7 is only my belief. Whether John is tall or not is still not fuzzy at all. Hence fuzzy set theory and probability serve difference purposes.

There are two approaches in pattern recognition—the *decision-theoretic* and *structural* approach. In decision-theoretic approach, the pattern is represented by a set of "features", or numerical values, which forms a feature vector. The classification is then performed in the feature space, using either a discriminant function, or *Bayes classifier* [3], [4].

In structural approach, the object is decomposed into a set of pattern primitives and a structure relating these primitives. There are two main structures that are commonly used in pattern recognition—phrase structured grammars and graphs. In the former case, a pattern is represented as a string of pattern primitives. For each pattern class, a phrase structured grammar, usually a context-free grammar, is constructed through a process of grammatical inference [5]. If the pattern string is in the language of the grammar, the pattern is recognized [6]. However, the string representation is essentially a linear structure. In order to represent a non-linear structure, one need resort to either a high-dimensional grammar, such as tree grammar or web grammar, or a *Guarded Grammar* [7], which can fulfill most function of the high dimensional grammar while retaining simplicity.

Another alternative to represent pattern structures is by the use of graphs. A graph consists of vertices representing pattern primitives, and edges representing relations between the primitives. The graph representation provide much richer structural content than grammatical approach. In order to incorporate more information into the data structure, Tsai and Fu [8] proposed to use attributed graph. Each pattern is represented by an attributed graph which is then matched against a template graph by either graph isomorphism [8] or graph monomorphism algorithms [9], [10]. Attributed graph was further extended to *Random Graphs* by including probabilistic information [11], [12]. However, the attributed graph proposed do not contain any fuzzy information. Random Graphs only deal with probabilistic uncertainty and cannot cope with fuzzy concepts which commonly appears in real-life applications. Hence the author proposed an extension, which is called *Fuzzy-Attribute Graph* (*FAG*) [1]. Some nice properties of FAG has also been derived.

The method that was presented in [1] relies on the existence of template graphs that is used for the matching. Instead of asking the user to provide these template graphs, we would like to have a learning algorithm that can derive the template graphs from a set of training patterns. This is parallel to the grammatical inference process in syntactical pattern recognition. In this paper, we provide an algorithm for learning templates from several example patterns, each of them are in the form of FAGs, and hence present a complete framework, including both training and recognizing phases, that can be used in structural pattern recognition.

II. Fuzzy-Attribute Graph

Attributed graph was introduced by Tsai and Fu [8] for pattern analysis. It provides a more straightforward representation of structural patterns. The vertices of the graph represent pattern primitives describing the pattern while the arcs are the relations between these primitives. However the pattern often possess properties that are fuzzy in nature and it has been extended to include such fuzzy information.

In a fuzzy-attribute graph, each vertex may have attributes from the set $Z = \{z_i | i = 1, \cdots, I\}$. For each attribute z_i, it may take values from $S_i = \{s_{ij} | j = 1, \cdots, J_i\}$. The set of all possible attribute-value pairs is $\tilde{L}_v = \{(z_i, \tilde{A}_{S_i}) | i = 1, \cdots, I\}$ where $\tilde{A}_{S_i}$ is a fuzzy set on the attribute-value set S_i. A valid pattern primitive is just a subset of $\tilde{L}_v$ in which each attribute appears only once, and $\tilde{\Pi}$ represent the set of all those valid pattern primitives.

Similarly, each arc may have attributes from the set $F = \{f_i | i = 1, \cdots, I'\}$ in which each f_i may take values from $T_i = \{t_{ij} | j = 1, \cdots, J_i'\}$. $\tilde{L}_\alpha = \{(f_i, \tilde{B}_{T_i}) | i = 1, \cdots, I'\}$ denotes the set of all possible relational-attribute value pair, where $\tilde{B}_{T_i}$ is a fuzzy set on the relational attribute-value set T_i. A valid relation is just a subset of $\tilde{L}_\alpha$ in which each attribute appears only once. The set of all valid relation is denoted $\tilde{\Theta}$.

Each fuzzy-attribute graph contains an underlying graph structure H, represented by the vertices and edges, and the attributes for them, $\tilde{L}_v$ and $\tilde{L}_\alpha$. Hence we can define a FAG as follows:

Definition 1: A *Fuzzy-Attribute Graph* (*FAG*), $\tilde{G}$ over $\tilde{L} = (\tilde{L}_v, \tilde{L}_\alpha)$ with an underlying graph structure $H = (N, E)$ is defined to be an ordered pair $(\tilde{V}, \tilde{A})$, where $\tilde{V} = (N, \tilde{\sigma})$ is called a fuzzy vertex set and $\tilde{A} = (E, \tilde{\delta})$ is called a fuzzy arc set and

$\tilde{\sigma}\colon N \to \tilde{\Pi}$ is called a fuzzy vertex interpreter,

$\tilde{\delta}\colon E \to \tilde{\Theta}$ is called a fuzzy arc interpreter.

This definition also applies when there are non-fuzzy attributes, since a crisp (non-fuzzy) set can always be represented as a special case of a fuzzy set.

Example 1: When we use *FAG* to represent Chinese characters, each vertex of the graph will represent a stroke, and the relation between the vertices are represented by the edges. Each stroke may have, say, two different attributes *STROKE-TYPE* and *LENGTH.* The attributes for relations between the nodes are *JOINT-TYPE, VERT-REL* and *HORI-REL,* for illustration. The values that can be taken in each of the attributes are,

$$\begin{aligned}\{STROKE\text{-}TYPE\} &= \{Vertical, Horizontal,\\ &\qquad Slant45, Slant135\}\\ \{LENGTH\} &= \{Long, Short\}\\ \{JOINT\text{-}TYPE\} &= \{T\text{-}from, T\text{-}into, Ht, Cross,\\ &\qquad Parallel\}\\ \{VERT\text{-}REL\} &= \{On\text{-}top\text{-}of, Below\text{-}of,\\ &\qquad No\text{-}vert\text{-}relate\}\\ \{HORI\text{-}REL\} &= \{Left\text{-}of, Right\text{-}of,\\ &\qquad No\text{-}hori\text{-}relate\}.\end{aligned}$$

Using these attributes and values, we can construct a *FAG* to represent a Chinese character. A typical fuzzy vertex set and fuzzy arc set is given below,

$$\begin{aligned}\tilde{\pi}_1 = \{&(STROKE\text{-}TYPE, \{0.7/Vertical,\\ &0.85/Slant45, 0.01/Horizontal, 0/Slant135\}),\\ &(LENGTH, \{0.6/Long, 0/Short\})\}\end{aligned}$$

and,

$$\begin{aligned}\tilde{\theta}_1 = \{&(JOINT\text{-}TYPE, \{0.7/T\text{-}from, 0.65/Cross,\\ &0/T\text{-}into, 0/Ht, 0/Parallel\}),\\ &(VERT\text{-}REL, \{0.9/On\text{-}top\text{-}of, 0/Below\text{-}of,\\ &0.25/No\text{-}vert\text{-}relate\}),\\ &(HORI\text{-}REL, \{0.2/Left\text{-}of, 0.4/Right\text{-}of,\\ &0.77/No\text{-}hori\text{-}relate\})\}.\end{aligned}$$

$\tilde{\pi}_1$ represents a stroke which is quite long, and can be interpreted as either a vertical stroke or a upward slant stroke with the respective membership values. Similarly, $\tilde{\theta}_1$ represents either a cross relation or a T-joint relation between the corresponding strokes (uncertainty exists because for example, if a user writes down the vertical stroke of a T-joint a little bit longer, it becomes a cross joint).

The *FAG* data structure is general enough to be applied to most patterns in typical recognition problems. The fuzzy attributes are used to handle those fuzzy properties as well as non-fuzzy enumerable attributes. For those attributes with continuous values, such as height of a man, we can easily abstract out the fuzzy concepts such as *tall* and *short,* and represent them in fuzzy sets. This is what is usually performed by human being.

III. A Recognition Framework Based on *FAG*

In this section, we will describe a framework of general pattern recognition system based on the *FAG.* Before going further, the author would like to reiterate his view that human perception is based on recognizing certain identifiable parts from a complex scene. Based on these identified parts, he can then reconstruct the whole picture. Also, the *ideal* patterns, or the *definition* of the pattern is not fuzzy at all. It is an *instance* of this definition that is fuzzy. The whole approach is based on these views and assumptions.

The following is a generalization of the approach from the author's original application on Chinese character recognition [1]. First of all, we define a set of small subpatterns (similar to syntactic pattern recognition). The definition of these subpattern is represented in a Hard (non-fuzzy) *FAG* or *HFAG,* which is a *FAG* with all fuzzy set crisped, i.e. with membership of either 0 or 1. These are the "ideal" patterns and are used as the matching templates. The pattern to be matched is represented as a *FAG* because each instance of the pattern is fuzzy. The fuzzy graph is then matched against the non-fuzzy graph to determine whether a matching is found. One problem that immediately follows is the way to define a matching.

To derive the matching formula, we first notice that the *FAG* representation has some meaning behind it. Actually, the membership value of a fuzzy set can be interpreted as the compatibility of an object and its attributes. Consider the following examples of a fuzzy vertex set,

$$\tilde{A}_{1S_i} = \{\mu_{\tilde{A}_{1S_i}}(s_{ij})/s_{ij} | s_{ij} \in S_i\}$$

where S_i is the ith attribute and s_{ij} are the attribute values. This can be interpreted as

$$\text{vertex } v_i \text{ possesses property } s_{ij}$$
$$\text{with truth value } \mu_{\tilde{A}_{1S_i}}(s_{ij}).$$

To match between the template graph and the pattern graph, one would like to check the truth value of the following statement:

$$\text{template } \tilde{H} \text{ possesses property } s_{ij} \; AND$$
$$\text{pattern } \tilde{G} \text{ possesses property } s_{ij}$$

and this can be obtained by the following fuzzy expression

$$\mu_{\tilde{H}_{S_i}}(s_{ij}) \wedge \mu_{\tilde{G}_{S_i}}(s_{ij}).$$

For the same attribute S_i, we would like to take *disjunction* over all attribute values.

$$\text{template } \tilde{H} \text{ possesses property } s_{i1} \; AND$$
$$\text{pattern } \tilde{G} \text{ possesses property } s_{i1} \; OR$$
$$\text{template } \tilde{H} \text{ possesses property } s_{i2} AND$$
$$\text{pattern } \tilde{G} \text{ possesses property } s_{i2} \; OR \ldots$$

This is for one attribute and we can repeat this for all attributes by taking *conjunction* over them. Hence we have the following definition:

Definition 2: Let $\tilde{G}_1$ and $\tilde{G}_2$ be two *FAGs* with underlying graph $H_1 = (N_1, E_1)$ of $\tilde{G}_1$ being monomorphic to the underlying graph $H_2 = (N_2, E_2)$ of $\tilde{G}_2$. The *degree of matching* γ is defined as

$$\gamma(\tilde{G}_1, \tilde{G}_2) = \bigwedge_{i \in N_1} \alpha(i, h(i)) \cdot \bigwedge_{(i,j) \in E_1} \beta(e_1(i,j), e_2(h(i), h(j)))$$

where $h(i)$ is the vertex in $\tilde{G}_2$ that matches vertex i in $\tilde{G}_1$, and $e_k(i,j)$ is the arc joining vertices i and j in $\tilde{G}_k$.The value $\alpha(i,j)$ is the matching between vertex i and vertex j and is obtained as

$$\alpha(i,j) = \bigwedge_{m=1}^{I} \bigvee_{n=1}^{J_I} \{(\mu_{\tilde{A}_{1S_m}}(s_{mn}) \wedge \mu_{\tilde{A}_{2S_m}}(s_{mn}))\}$$

where $\tilde{A}_{kS_i}$ is the fuzzy set of the attribute S_i of graph $k, k = 1, 2$.

Similarly, the value $\beta(e_i, e_j)$ is the matching between arc e_i and arc e_j and is obtained as

$$\beta(e_1, e_2) = \bigwedge_{m=1}^{I'} \bigvee_{n=1}^{J'_m} \{(\mu_{\tilde{B}_{1T_m}}(t_{mn}) \wedge \mu_{\tilde{B}_{2T_m}}(t_{mn}))\}$$

where $\tilde{B}_{kT_i}$ is the fuzzy set of the attribute T_i of graph $k, k = 1, 2$.

Definition 3: Let $\tilde{G}_1$ and $\tilde{G}_2$ be two FAGs. $\tilde{G}_1$ is $\boldsymbol{\lambda}$-*monomorphic* to $\tilde{G}_2$ if

1) the underlying graph H_1 of $\tilde{G}_1$ is monomorphic to the underlying graph H_2 of $\tilde{G}_2$, and
2) the degree of matching $\gamma(\tilde{G}_1, \tilde{G}_2) \geq \lambda$.

The degree of matching so defined has its physical meaning which corresponds to the logical expression above. This logical expression is both intuitive and understandable and in fact it has some very nice property that makes it very useful in pattern recognition. Please refer to [1] for more detail.

IV. Learning From Examples of *FAGs*

In the previous section, we have discussed how matching can be achieved between a fuzzy instance and the non-fuzzy definition. This definition (the "ideal" pattern) was provided by human experts. However, we would like to develop a training algorithm such that the definition templates can be derived from a set of fuzzy examples automatically by computer. In this section, we will try to describe such an algorithm. This is not a trivial "learning from example" problem as the examples are fuzzy in nature, and the result should be λ-monomorphic with all the fuzzy examples.

Theorem 1: Given a *FAG* $\tilde{G}$, and a *HFAG* $\tilde{H}, \tilde{G}$ is λ-monomorphic with $\tilde{H}$ iff for each nodal attribute $z_i, \exists$ distinct $s_{ij} \in S_i$ such that $\mu_{\tilde{G}_{S_i}}(s_{ij}) \bigwedge \mu_{\tilde{H}_{S_i}}(s_{ij}) \geq \lambda$ and for each relational attribute $f_i, \exists$ distinct $t_{ij} \in T_i$ such that $\mu_{\tilde{G}_{T_i}}(t_{ij}) \bigwedge \mu_{\tilde{H}_{T_i}}(t_{ij}) \geq \lambda$.

This follows directly from the definition of λ-monomorphic.

This theorem, although simple, allows us to consider each attribute (whether it is nodal or relational) individually. Hence the following discussion will be concentrated on the learning of a single attribute, and the result can then be combine together back to form the template graph.

Definition 4: Consider a set $X = \{x_i | i = 1, \cdots, n\}$. The extended product $\odot$ between two fuzzy subset $\tilde{Y}_1$ and $\tilde{Y}_2$ of X is defined as

$$\tilde{Y}_1 \odot \tilde{Y}_2 = \bigvee_{i=1}^{n} (\mu_{\tilde{Y}_1}(x_i) \wedge \mu_{\tilde{Y}_2}(x_i)).$$

Definition 5: $\tilde{Y}_1$ and $\tilde{Y}_2$ are said to be λ-matched if $\tilde{Y}_1 \odot \tilde{Y}_2 \geq \lambda$.

With the help of Theorem 1 we can now rephrase the problem to a simpler form: Consider a set $X = \{x_i | i = 1, \cdots, n\}$. Given a set of fuzzy subset of $X, \tilde{Y}_i, i = 1, \cdots, m$, we are going to find the smallest set $Z \subseteq X$ (i.e. $|Z|$ is smallest) such that $\tilde{Z}$, the fuzzy set representation of Z, is λ-matched with $\tilde{Y}_i, i = 1, \cdots, m$. This is for one attribute and we repeat this for all nodal and relational attributes.

Definition 6: Given a universe $X = \{x_i | i = 1, \cdots, n\}$. A *polynomial form* P is a summation of the form

$$P = \sum_{k=1}^{m} a_k X_k$$

where X_k is a product of any number of x_i's, and a_k are constants.

For each fuzzy set $\tilde{Y}$ on the universe X, we can rewrite $\tilde{Y}$ in the polynomial form

$$\sum_{i=1}^{n} \mu_{\tilde{Y}}(x_i) x_i$$

where each X_i's is a singleton.

Definition 7: The *f-product* between two polynomial form P_1 and P_2 is another polynomial form $Q = P_1 \otimes P_2$ such that if $P_1 = \Sigma_{k=1}^{m} a_k X_k$ and $P_2 = \Sigma_{i=1}^{m'} a'_k X'_k$, then

$$Q = \sum_{i=1}^{m} \sum_{j=1}^{m'} \min(a_k, a'_k) X_i X_j$$

with the convention that $x_i x_i = x_i$, and

$$a_1 X + a_2 X = \max(a_1, a_2) X.$$

Example 2:

$$\begin{aligned}
&(0.5x_1 + 0.3x_2 + 0.7x_3) \otimes (0.3x_1 + 0.2x_2 + 0.8x_3) \\
&\quad = (0.3x_1 + 0.2x_2 + 0.7x_3) + (0.2x_1x_2 + 0.3x_1x_2) \\
&\qquad + (0.5x_1x_3 + 0.3x_1x_3) + (0.3x_2x_3 + 0.2x_2x_3) \\
&\quad = 0.3x_1 + 0.2x_2 + 0.7x_3 + 0.3x_1x_2 + 0.5x_1x_3 \\
&\qquad + 0.3x_2x_3.
\end{aligned}$$

Theorem 2: Let $\tilde{Y}_1$ and $\tilde{Y}_2$ be two fuzzy sets with polynomial forms P_1 and P_2 respectively, and Q be the *f-product*

$$Q = P_1 \otimes P_2 = \sum_{k=1}^{m} a_k X_k.$$

Then the *HFAG* $\tilde{H}_k$ constructed form the kth term $a_k X_k$ such that

$$\mu_{\tilde{H}_k}(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is in } X_k \\ 0 & \text{otherwise} \end{cases}$$

is a_k-matched with both $\tilde{Y}_1$ and $\tilde{Y}_2$.

Proof: Note that P_1 and P_2, the polynomial form of $\tilde{Y}_1$ and $\tilde{Y}_2$ contain only singleton x_i's in the summation. Let $P_1 = \Sigma_{i=1}^{n} a_i x_i$ and $P_2 = \Sigma_{i=1}^{n} a'_i x_i$. Then for a term containing $x_i x_j$ in the product (note that $x_1 = x_1 x_1$) with a coefficient b_{ij},

$$b_{ij} = \max(\min(a_i, a'_j), \min(a_j, a'_i)).$$

Hence,

$$\begin{aligned}
&(\mu_{\tilde{H}_k}(x_i) \wedge \mu_{\tilde{Y}_1}(x_i)) \bigvee (\mu_{\tilde{H}_k}(x_j) \wedge \mu_{\tilde{Y}_1}(x_j)) \\
&\quad = \mu_{\tilde{Y}_1}(x_i) \vee \mu_{\tilde{Y}_1}(x_j) \\
&\quad = \max(a_i, a_j) \geq b_{ij}.
\end{aligned}$$

The same argument applies to $\tilde{Y}_2$. Hence the result.

It can easily be seen that the above argument can be extended to more than two $\tilde{Y}$'s. From Theorem 1 we can separate the learning into individual attributes for nodes and relations. Theorem 2 allows us to find the attribute values for individual attribute. It can further be noted that we can delete intermediate terms which has a value $<\lambda$.

Definition 8: The λ-cut of a polynomial form $P = \Sigma_{i=1}^{n} a_i X_i$ is another polynomial form $P_\lambda = \Sigma_{i=1}^{n} b_i X_i$ such that

$$b_i = \begin{cases} a_i & \text{if } a_i \geq \lambda \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 3: The λ-cut operation is distributive with respect to *f-product,* i.e.

$$P_{1\lambda} \otimes P_{2\lambda} = (P_1 \otimes P_2)_\lambda.$$

From Definition 7, it can be seen that any term that contains a coefficient $<\lambda$ will produce another term which is $<\lambda$ in the summation. This will be discarded after the λ-cut operation. Hence we can ignore any terms that contain coefficients $<\lambda$.

With this theorem, we can delete all intermediate terms with coefficient $<\lambda$ during the multiplication process. This will prune off all those unpromising terms. Taking this into account, we have the following algorithm for finding the "smallest" templates.

Algorithm:

Repeat the following steps for each nodal and relational attributes.

1) Represent the fuzzy set of the attribute of all training samples in a polynomial form, namely, $P_1, P_2, \cdots, P_n$.
2) Assign λ-cut of P_1 to Q.
3) Find the *f-product* $Q \leftarrow \lambda$-cut of $(Q \otimes P_2)$.
4) Repeat step 3 until all input samples exhausted.
5) Find the term with smallest number of x_i's and with coefficient $\geq \lambda$.

Finally, construct a *HFAG* from these term.

The complexity of this algorithm is $O(nm2^m)$ for each attribute where n is the number of examples and m is the number of attribute-values within the attribute. For each multiplication, one of them is originally a *FAG* and its polynomial form contain only m terms. The other have at most 2^m terms. Hence each step take no more than $m2^m$. Also note that the value of m is fixed and is usually very small, for example ≤ 10. In our previous example, the *STROKE-TYPE* attribute actually contain only 4 values, i.e. $m = 4$ and $m2^m = 64$. Hence the algorithm is essentially linear with respect to the number of training examples. Finding the smallest term is also constant (in this case $O(2^m)$).

V. An Illustrative Example

Chinese character recognition is a typical example that require fuzziness in the description of the pattern primitives. Each stroke and their relations can have fuzzy attribute as described in Example 1 of Section II. To test the effectiveness of the algorithm we try the approach on the character ±. Fig. 1 shows 10 samples of the character ±. For simplicity, we only use 1 attribute—the stroke type, for illustration. The stroke type may take four values—*Horizontal, Vertical, Slant45* and *Slant135,* which are the main direction of strokes in a Chinese characters (two in horizontal and vertical and two in the diagonals). They are represented as a fuzzy set. Table I shows the membership values of the fuzzy set *STROKE-TYPE.*

The method of finding the membership values can be found in [13]. First of all, each character is represented by a *FAG.* The *FAGs* are first matched against each other to find vertex correspondence. This can be achieved by a graph matching algorithm, such as that proposed by A. K. C. Wong [9]. After the vertex correspondence are found, the algorithm can then be applied. When applying the above algorithm, taking $\lambda = 0.25$, we will get the following result:

Let H, V, S and X represent the attribute-values *Horizontal, Vertical, Slant45* and *Slant135* respectively, and Q_1, Q_2 and Q_3 be the resultant polynomial form for the three strokes in the character ±. We get

$$\begin{aligned} Q_1 &= 0.37H + 0.49HX + 0.37HS + 0.39HSX \\ Q_2 &= 0.3V + 0.39VS \\ Q_3 &= 0.31H + 0.31HS + 0.61HX + 0.61HSX. \end{aligned}$$

Taking the term with the smallest number of attributes, then

$$\begin{aligned} \text{Stroke } 1 &= Horizontal \\ \text{Stroke } 2 &= Vertical \\ \text{Stroke } 3 &= Horizontal. \end{aligned}$$

This definition is the same to what we would expect.

When we matched this template with the original 8 patterns, the degree of matching are respectively 0.85, 0.83, 0.37, 0.43, 0.31, 0.84, 0.81, and 0.30, all exceed our threshold $\lambda = 0.25$. Hence correct result is produced by the algorithm.

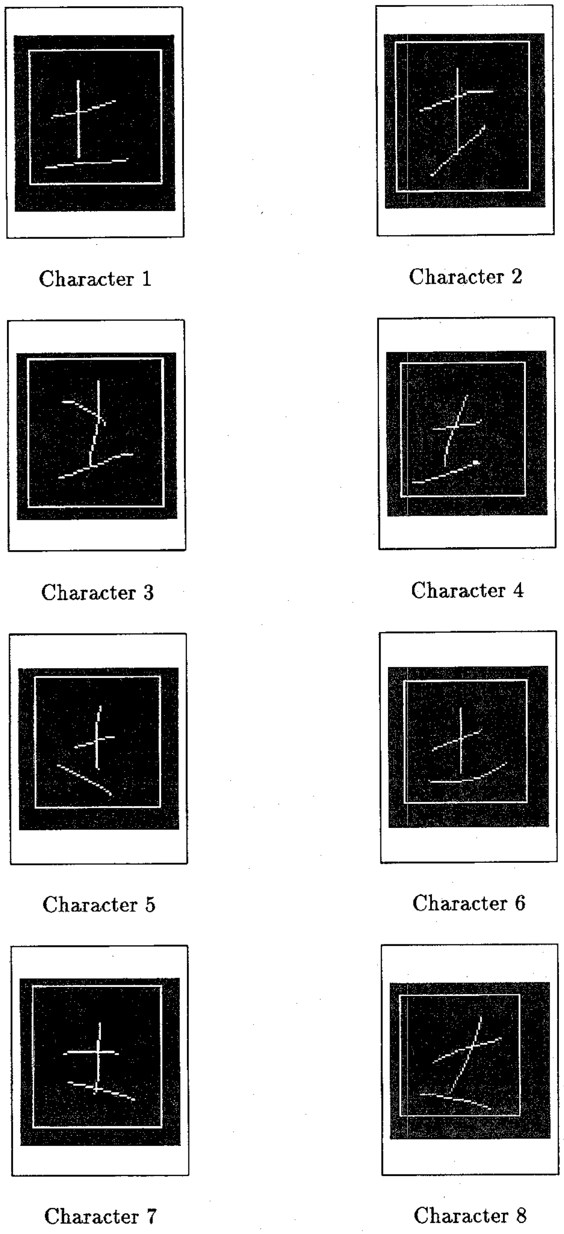


Fig. 1. A set of training samples used.

VI. Conclusion

In this paper, we have discussed a learning algorithm that can derive a representation of pattern templates from a set of fuzzy examples. This can be applied in pattern recognition approach based on the *Fuzzy-Attribute Graph.* It is the author's belief that human perception is performed by looking for identifiable subpatterns in a complex scene and based on these subpatterns, make decision about the scene. Hence we approach the problem by finding subpatterns using matching of *FAG*s and then combine the matched ones to form the pattern.

Also the template used for matching is a kind of definition which should not contain any fuzziness, while any "instance" of the template is fuzzy. Hence we define the matching between a fuzzy instance and a non-fuzzy template. However, we do not want to define the template by the users. This will be very tedious and error prone, especially when we have huge number of templates to define, such as in Chinese character recognition. We would like to have an automatic mechanism to do this instead.

The algorithm that is proposed can find the smallest template from a set of training samples based on the matching metric. We have proved that the resultant *FAG* from the algorithm

TABLE I
MEMBERSHIP VALUES OF THE FUZZY SET STROKE-TYPE

Sample No.	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.85	0.00	0.22	0.00
2	0.83	0.00	0.19	0.00
3	0.37	0.00	0.00	0.49
4	0.91	0.00	0.13	0.00
5	0.84	0.00	0.21	0.00
6	0.98	0.00	0.39	0.00
7	1.00	0.00	0.03	0.00
8	0.94	0.00	0.31	0.00

(a) Stroke 1

Sample No.	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.00	1.00	0.03	0.03
2	0.00	1.00	0.03	0.03
3	0.00	0.93	0.09	0.00
4	0.00	0.43	0.32	0.00
5	0.00	0.99	0.05	0.02
6	0.00	1.00	0.03	0.03
7	0.00	0.95	0.08	0.00
8	0.10	0.30	0.39	0.00

(b) Stroke 2

Sample No.	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.96	0.00	0.09	0.00
2	0.86	0.00	0.94	0.00
3	0.89	0.00	0.25	0.00
4	0.85	0.00	0.22	0.00
5	0.31	0.00	0.00	0.61
6	0.84	0.00	0.18	0.00
7	0.81	0.00	0.00	0.20
8	0.85	0.00	0.00	0.18

(c) Stroke 3

is in fact λ-monomorphic to all the learning pattern. As an illustration, the algorithm was applied to 8 character 士 and the above verified. The learned result also matches with our daily experience.

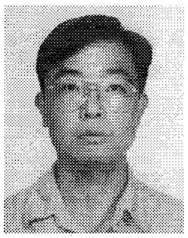
Finally, the algorithm is linear on the number of training samples. Although it is exponential with respect to the number of values in each attribute, this is usually very small. In our application on Chinese character recognition, the largest number is 6. Hence this can be assumed to be a constant. Thus, an efficient algorithm has been developed for the learning of *HFAG* from a set of examples of *FAGs*.

ACKNOWLEDGMENT

This project is supported by the Hong Kong Research Grant Council.

REFERENCES

[1] K. P. Chan and Y. S. Cheung, "Fuzzy-attribute graph with application to Chinese character recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 4, pp. 402–410, Mar. 1992.
[2] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.
[3] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.*Reading, MA: Addison-Wesley, 1977.
[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[5] K. S. Fu and T. L. Booth, "Grammatical inference: Introduction and survey part I and II," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-5, pp. 59–72, 409-P23, Jan./July 1975.
[6] K. S. Fu, *Syntactic Pattern Recognition and Applications.* New York: Prentice Hall, 1982.
[7] K. P. Chan, "Guarded fuzzy-attribute context free grammar and its application on structural pattern recognition," *Int. J. Pattern Recognition and Artificial Intelligence*, vol. 6, no. 5, 1992.
[8] W. H. Tsai and K. S. Fu, "Error correcting isomorphism of attributed relational graphs for pattern analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 12, pp. 757–768, Dec. 1979.
[9] A. K. C. Wong, M. You, and S. C. Chan, "An algorithm for graph optimal monomorphism," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 3, pp. 628–638, May 1990.
[10] F. A. Akinniyi, A. K. C. Wong, and D. A. Stacey, "A new algorithm for graph monomorphism based on the projections of the product graph," *IEEE Trans. Syst., Man, Cybern.*, vol. 16, no. 5, pp. 740–751, Sept. 1986.
[11] A. K. C. Wong and D. E. Ghahraman, "Random graphs: Structural-contextual dichotomy," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 2, no. 4, pp. 341–348, July 1980.
[12] A. K. C. Wong and M. You, "Entropy and distance of random graphs with application to structural pattern recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 7, no. 5, pp. 599–609, Sept. 1985.
[13] K. P. Chan, "Fuzzy-theoretic approach to handwritten Chinese character recognition," University of Hong Kong, Ph.D. thesis, 1989.
[14] ——, "Learning templates from fuzzy examples in structural pattern recognition," *Proc. 3rd IEEE Int. Conf. on Fuzzy Systems*, vol. 2, pp. 608–613, June 1994.



Kwok-Ping Chan (M'88) received the B.Sc. (Eng.) and Ph.D. degrees in electrical and electronic engineering from the University of Hong Kong in 1984 and 1989, respectively.

He has taught at Hong Kong Polytechnic, University of Hong Kong and City Polytechnic of Hong Kong. He is currently a lecturer in the Department of Computer Science, University of Hong Kong. His current research interest is in pattern recognition, image processing, and fuzzy set theory.