Chapter 7

FSEAC: Fuzzy Simple and Effective Associative Classifier

Most real-life datasets are neither only binary nor only numerical but a combination of both. The general method (as discussed in the introduction – Chapter 1) adopted is to convert numerical attributes into binary attributes using ranges. This reduces the problem to traditional ARM and classification with binary values. But this approach introduces a loss of information, and also increases the uncertainty in the data. Sharp partitions also give rise to polysemy (one partition with two or more distinct meanings) and synonymy (two or more partitions having the same meaning), with small changes in intervals of the sharp partitions generating misleading results. Moreover, the intervals do not generally have clear semantics associated.

A better way to solve this problem is to have attribute values represented in the interval $[0, 1]$, instead of having just 0 and 1, and to have transactions with a given attribute represented to a certain extent (in the range $[0, 1]$). In this way crisp binary attributes are replaced by fuzzy ones. Doing so ensures that there is no loss of information whatever the value of any numerical attribute. Moreover, the inherent uncertainty that is present in numerical data is also appropriately taken care of.

Thus, in this chapter we present Fuzzy Simple and Effective Associative Classifier (FSEAC) which extends the crisp Associative Classifier SEAC, while preserving its advantages, such as simplicity and ease-of-use, along with comprehensible rule sets. It includes a number of modifications and extensions. FSEAC amalgamates Fuzzy Logic with Associative Classification. A key feature of FSEAC, in comparison with SEAC, is its ability to represent different facets of uncertainty involved in a classification decision in a more faithful way through the use of Fuzzy Logic.

Like SEAC, FSEAC uses the actual association rules also for the final classification process. And, the two major hurdles in using these association rules directly, namely their huge volume and redundancy are dealt with appropriately by FSEAC. It uses a constrained exhaustive approach (detailed in Section 7.2.5) in order to obtain the best possible results in terms of accuracy, and does not use any greedy approach. FSEAC deals with redundant association rules through an effective and simple pruning technique which also helps in cutting down the number of rules which finally form a part of the classifier.

We have come up with a new information metric called Fuzzy-weighted Information gain which is a fuzzy adaptation of Information gain. Fuzzy-weighted Information gain takes the fuzzy membership representation (using a t-norm) of each Classification Association Rule (CAR) into account. It is used for pruning the CAR set, and also for scoring unlabeled test data tuples.
On the parameter configuration front, FSEAC is as easy to configure as SEAC is. The only two parameters (best \(k\%\) rules and \(\text{minlen}\)) needed for the classifier building process, are simple to configure and require very little effort from the user. Well-known parameters used for ARM, like support and confidence, are still used, but only when mining rules, and not while building the actual classifier.

A high-level overview of the steps followed for fuzzy classifier generation using FSEAC are:

- Pre-processing using FPreP
- CARs mining using FAR-Miner
- CARs pruning based on Fuzzy-weighted Information Gain and rule length

Overall, FSEAC is a superior algorithm as compared to SEAC, especially in terms of accuracy. This is illustrated and analyzed in more details in Section 7.5. The salient features of FSEAC are:

- It uses soft partitions through fuzzy clustering. Soft partitions help in dealing with numerical attributes in an effective manner.
- Rule pruning is done based on rule-length and Fuzzy-weighted information gain
- It uses a constrained exhaustive approach unlike the greedy approach taken by other associative classifiers. It is thus able to explore a larger space, yielding better accuracy.
- FSEAC directly mines CARs, instead of using additional algorithms like FOIL and PRM, thus making classifier training much simpler and faster.

### 7.1 Problem Definition

We use the following notation:

- Let \(D\) be the original dataset.
- \(E\) = transformed version of \(D\) with fuzzy attributes
- \(I\) = set of all items in \(E\)
- \(J\) = set of all frequent itemsets in \(E\)
- \(C\) = set of class labels (consequent of a rule) = \(\{y_1, y_2, \ldots, y_m\}\)
- \(S\) = set of class labels singled out for local ARM processing
- \(\{A_1, A_2, \ldots, A_k\}\) = set of all possible attributes in \(E\)
- \(p\) = literal, which is an attribute-value pair \((A_i, v)\) where \(A_i\) is an attribute and \(v\) is its corresponding value
- \(X\) = precedent of a rule \(\forall X \equiv p_1, p_2, \ldots, p_j\) and \(X \subseteq I\)
• $R$ = set of all rules used for classification = \{r_1, r_2, \ldots, r_n\}

• $t$ = tuple $\forall t \in E$

• $c$ = number of fuzzy clusters/partitions per numerical attribute generated through fuzzy clustering

A tuple $t$ satisfies a literal $p = (A_i, v)$ iff $\tau_i = v$, where $\tau_i$ is the value of the $i_{th}$ attribute of $t$. A rule $r$ takes the form $X \rightarrow Y$, where $X$ is the precedent of the rule and can take on any itemset $x$ as its value. $Y$ is the antecedent of the rule and can take on any value $y \in C$. $t$ satisfies rule $r$’s body iff it satisfies every literal in the rule. If $t$ satisfies $r$’s body, $r$ predicts that $t$ is of class $y$.

The fuzzy training dataset $E$ is derived for the original training dataset $D$ through fuzzy pre-processing. And, fuzzy association rules are mined from $E$ using an appropriate mining algorithm.

The rule set $R$ is formed while building the classifier from the association rules. And, the final classification, of any tuple with unknown class label, is based on a function of best $k\%$ rules of any class label $y$ satisfying the tuple and the rule length cut off $l$ of each of these rules – $f(k, l)$.

### 7.2 FSEAC and Rule Generation

FSEAC has been designed to directly use the association rules generated by an ARM algorithm like FAR-Miner, instead of creating rules using various other techniques. And, in the process we have ensured that huge number of association rules does not negatively impact the final classification process. This is achieved by pruning which brings down the rule set ($R$) size to an optimal value, which need not necessarily be very small as was assumed in previous algorithms like CBA, CMAR, and CPAR. The rule set size can be small to medium sized.

FSEAC works in a two-phased manner. The first phase generates association rules based on each class attribute $y \in C$ and based on the whole dataset $E$. Based on these association rules (after appropriate pruning), local association rule generation is done for each class attribute $y \in C$ which has very low representation or no representation at all in the association rules generated in the first phase. The final rule set $R$ is subsequently pruned in the same manner as in the first phase.

#### 7.2.1 Pre-processing and Fuzzy Association Rule Generation

Attributes can be binary or numerical. The fuzzy training dataset $E$ (derived for the original training dataset $D$) is a set of transactions, each of which is labeled with one of the $l$ classes. This transformation of the original crisp dataset to its fuzzy version is done through the fuzzy pre-processing (FPrep) described in Chapter 3 and [MP10b]. First Fuzzy c-Means (FCM) is applied on each numerical attribute in $D$ to obtain the corresponding fuzzy partitions/clusters. The number of fuzzy clusters/partitions per numerical attribute ($c$) is used during the FCM process.

A sample fuzzy training dataset is illustrated in Fig. ?? Each record has many attributes with each attribute ($a$) having a corresponding value ($v$) and a fuzzy membership ($\mu$) – $\langle a, v, \mu \rangle$. $\mu$ has a range $0 \geq \mu \geq 1$. In case of a fuzzy attribute, $v$ is a fuzzy partition associated with the attribute, and $\mu$ is the
corresponding fuzzy membership (e.g. \( \text{Age} = \text{Young}^{0.9} \)). For a binary attribute \( v \) is its value, with \( \mu \) taking the default value 1 (e.g. \( \text{Gender} = \text{Male}^{1} \)).

\[
\begin{align*}
  a_1 &= v_{1,1}^{\mu_{1,1}}, a_2 = v_{1,2}^{\mu_{1,2}}, \ldots, a_k = v_{1,k}^{\mu_{1,k}}, \text{class\_label} \\
  \cdots \\
  a_1 &= v_{n,1}^{\mu_{n,1}}, a_2 = v_{n,2}^{\mu_{n,2}}, \ldots, a_k = v_{n,k}^{\mu_{n,k}}, \text{class\_label}
\end{align*}
\]

**Figure 7.1 Fuzzy Dataset Representation**

FAR-Miner (Chapter 4 and [MP09]) is used to derive fuzzy association rules from the whole dataset \( E \) (global rules) and from subsets of \( E \) pertaining to classes in \( C \) (local rules). Appropriate minimum support (\( \text{minsup} \)) and minimum confidence (\( \text{minconf} \)) are used for this mining process.

### 7.2.2 First phase

In this section, we describe the first phase of FSEAC. We assume the following notations, in addition to the ones defined in Section 6.1:

- \( RCM \) = Hashmap like data structure having class labels (every \( y \in C \)) with their respective counts in \( R \)
- \( R_{IG} \) = Hashmap like data structure having rules (every \( r \in R \)) with their respective information gains (\( IGs \))
- \( S \) = set of all class labels that need to be processed locally in the second phase
- \( g_{max} \) = the maximum count (frequency) of all the counts for the classes (\( ys \))

In the first phase ARM is done throughout the dataset for all the consequent class labels (every \( y \in C \)), i.e. in a global manner. For this ARM process (Algorithm 14, line 5), user-specified values of support (\( \text{minsup} \)) and confidence (\( \text{minconf} \)) are used. Through this ARM process, an initial rule set \( R \) is formed.

Information gain which is one of the best metrics of information has been used for the classifier training process. The entropy and information gain is calculated for each rule in the rule set \( R \). Given a rule \( X \rightarrow Y \), \( X \) is the precedent of the rule and can take on any itemset \( x \) as its value. \( Y \) is the antecedent of the rule and can take on any value \( y \in C \). The entropy of \( Y \) is given by Eq. 10.6. The frequency (\( p_i \)) of \( y \) is simply discerned by checking if \( y \) exists in the \( i^{th} \) record (Algorithm 14, line 18–23). In general, \( n \) is the number of different values of \( y \) for which entropy is being calculated. In the case of FSEAC, \( n = 1 \) because only entropy for one particular class is calculated at one time.

The average conditional entropy \( H(Y|X) \) is given by Eq. 10.7. \( x \) is the itemset which forms the precedent of the rule. In the case of FSEAC \( X \) can take only one value which is \( x \), unlike the usual way of calculating average conditional entropy where all possible values that \( X \) can take are used. The presence of the class \( y \) in each record with itemset \( x \) is used in calculating \( H(Y|X) \). And, the
information gain $IG(Y|X)$ is given by Eq. 10.8. All the three equations, namely Equations 10.6, 10.7, and 10.8 are standard equations related to entropy and information gain.

$$H(Y) = - \sum_{i=0}^{z} p_i \log p_i$$ (7.1)

$$H(Y|X) = \sum \text{Prob}(X = x) H(Y|X = x)$$ (7.2)

$$IG(Y|X) = H(Y) - H(Y|X)$$ (7.3)

This is followed by the pruning step (Algorithm 15), which eliminates all redundant rules, thus drastically reducing the rule set size. The information gain $IG$ (Algorithm 15 line 5) of each rule is very important for the pruning process. Each rule $r_q$ is compared to all $r_{q+1}$ to $r_n$ rules (Algorithm 15 lines 6–24). And, in this comparison, for a given rule $r_q$, we see if there exists another rule $r_s$ such that both rules either have different consequents, or equal information gain (Algorithm 15 line 10). If $r_q \subset r_s$ (Algorithm 15 lines 11–16) then, $r_q$ is pruned ($R = R - r_q$), or else if $r_s \subset r_q$ (Algorithm 15 lines 17–21) then, $r_s$ is pruned ($R = R - r_s$). If both rules have the same information gain, then it means that they are formed from non-closed itemsets, and thus the smaller rule needs to be pruned. An itemset is not closed iff it has the same frequency as one of its supersets.

The pruning helps in both improving accuracy of FSEAC and in rule-set reduction. The first criterion of pruning, i.e. elimination of a rule if there is another rule which is its super-set and both rules have different class labels, helps in ensuring that only the best rules (both in terms of bigger rule-lengths and higher information gains) remain in the rule-set after the pruning is over. The ripple effect of this pruning criterion is that the rule-set size is also reduced considerably. According to the second, a rule is eliminated if there is another rule which is its superset, and both have same information gain and same class label, mainly helps in reducing the number of rules directly derived from closed itemsets. It also helps in improving accuracy to an extent. This reduction is considerable in datasets which have many non-closed itemsets.

After pruning, there is another iteration performed over the rule set $R$ to determine the counts of each consequent class label ($y \in C$). Then, the consequent class with maximum count is determined. Other consequent class labels are compared with this class label based on their respective counts, and those classes having counts less than a cut off value ($\text{cutOff} \times g_{max}$), i.e. having very low representation in $R$, are singled out. The default value for $\text{cutOff}$ is 0.5. Classes with no representation in $R$ are also singled out (Algorithm 14 lines 27–33). The ARM process is repeated for these consequent classes in a localized manner, i.e. for one class label at time. This local ARM process forms the second phase of FSEAC. If $\text{cutOff} \leq 1$, then local ARM is performed on all classes in the second phase of FSEAC.

### 7.2.3 Second phase

The second phase of FSEAC involves applying ARM individually to each of the class labels $y \in S$, i.e. those singled out for local ARM processing. The same minimum support $\text{minsup}$, as used in the first phase, is used for this local ARM. Separate datasets, one each pertaining exclusively to each class label $y \in S$, are created for this ARM process. These datasets are actually subsets of the dataset $E$. 

83
For a class $y$, the second phase of ARM is carried out on a subset $E_y \in \text{whole dataset } E$, such that $E_y$ contains all records of $E$ where $y$ is the class label. Thus, $E_y$ contains only those records of $E$ with class $y$. Local ARM, i.e., ARM only on records with class $y$ (subset $E_y$) is carried out with a local minimum support. Similar process is followed for other classes which are either under-represented or not represented at all in the rule set $R$ obtained after the first phase.

After iteratively doing ARM on each $y \in S$, the resultant rules are accumulated in $R'$ (Algorithm 16, lines 3–7). Then, a pass is made over the dataset $E$, in order to ascertain the count of the precedents of each rule in $R'$ (Algorithm 16, lines 8–25). These counts are calculated using the minimum fuzzy membership, in $t$, among the items involved in $X$. The counts are then used to calculate the entropy and information gain ($IG$) for each rule. All rules in $R'$ are then put in $R$, and their respective information gains are put in $R_{IG}$ (Algorithm 16, lines 26–32).

All rules in $R$ are pruned (Algorithm 15) in the same manner as in the first phase. This rule set $R$ is used for the actual classification as described in Section 7.3. Here, $PCM = \text{Hashmap like data structure}$ having precedents of rules (every $r \in R'$) with their respective counts in $E$.

### 7.2.4 Illustration of rule generation using FSEAC

In this section we detail two examples of how FSEAC can be used to train a classifier. The use of the classifiers thus built, is illustrated in section 7.3.

#### 7.2.4.1 Example 1

Table 7.1 shows the sample dataset that we use to illustrate the working of FSEAC. This dataset, containing binary ($C_b$) and numeric ($A_n$ and $B_n$) features, is transformed into a fuzzy dataset (Table 7.2) containing binary ($C_b$) and fuzzy ($A_f$ and $B_f$) features using the pre-processing technique (FPrep) described in Chapter 3 and [MP10b]. The fuzzy dataset is used for training.

For training the classifier, we first obtain all association rules (using $\text{mins} = 20\%$ and $\text{mincon} = 85\%$) for this dataset. The first phase, involving global ARM, yields 16 rules which are sorted according to information gain and then pruned (Table 7.3). The four rules that remain after pruning are put in the ruleset $R$. The dataset has two class labels, namely $X = 1$ and $X = 2$, with two rules each. With $\text{cutoff} = 50\%$, both classes are represented adequately, so the second phase is not executed. Thus, no addition of rules is done to $R$ (unpruned rules in Table 7.3), which becomes the final rule set, based on which classification can be done as illustrated in Section 7.3.1.1.

#### 7.2.4.2 Example 2

In the second example, we illustrate the training phases of FSEAC using a sample from the classifier trained for the UCI-ML dataset hepatic. In fact, FSEAC has been tested on various UCI-ML datasets, the details of which are in Section 7.5. The hepatic dataset has 155 records/tuples, of which 90% were used for training and 10% for testing.

For the first phase we used $\text{mins} = 30\%$ and $\text{mincon} = 85\%$ for the global ARM process. 211 rules (in rule set $R$) are generated, after appropriate pruning, in the first phase. The dataset has two
Table 7.1 Sample dataset for example 1

<table>
<thead>
<tr>
<th>$A_n$</th>
<th>$B_n$</th>
<th>$C_b$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.5</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1.6</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1.4</td>
<td>3.4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3.5</td>
<td>3.3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3.8</td>
<td>1.9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7.2 Sample dataset for example 1

<table>
<thead>
<tr>
<th>$A_f$</th>
<th>$B_f$</th>
<th>$C_b$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1*0.9668686</td>
<td>1*0.9995863</td>
<td>1*1</td>
<td>1</td>
</tr>
<tr>
<td>1*0.999974</td>
<td>1*0.96894675</td>
<td>2*1</td>
<td>1</td>
</tr>
<tr>
<td>1*0.997059</td>
<td>0*0.9707756</td>
<td>2*1</td>
<td>1</td>
</tr>
<tr>
<td>1*0.9117428</td>
<td>0*0.9707756</td>
<td>2*1</td>
<td>2</td>
</tr>
<tr>
<td>1*0.9984237</td>
<td>0*0.9933431</td>
<td>2*1</td>
<td>1</td>
</tr>
<tr>
<td>0*0.99502987</td>
<td>0*0.9806152</td>
<td>1*1</td>
<td>2</td>
</tr>
<tr>
<td>0*0.9953542</td>
<td>1*0.9337501</td>
<td>2*1</td>
<td>2</td>
</tr>
<tr>
<td>1*0.9668686</td>
<td>0*0.9707756</td>
<td>1*1</td>
<td>2</td>
</tr>
</tbody>
</table>

consequent class labels, namely $Class = 1$ and $Class = 2$. All 211 rules in $R$ have $Class = 2$ as their consequent, which means $Class = 1$ does not have any representation at all in $R$. Thus, $Class = 1$ is singled out for local ARM processing in the second phase and a dataset (subset of the hepatic dataset) is created which has all tuples with $Class = 1$.

For the local ARM processing of tuple with $Class = 1$, we use $\text{minsup} = 30\%$ (the same minimum support as used in the first phase). The rules thus generated are put in $R'$ and then pruned using the same strategy as in first phase. Then, all rules in $R'$ are integrated with $R$. The resultant rule set $R$ has representation for all possible consequent class labels, $Class = 1$ and $Class = 2$ in this case, and has 510 rules. This rule set can be directly used for classification as detailed in Section 7.3.1.

7.2.5 Constrained Exhaustive Approach of FSEAC

As mentioned in the introduction, FSEAC uses an constrained exhaustive approach to classifier building, which is one of the main reasons why FSEAC provides a very high degree of accuracy. The substructure of FSEAC is two-pronged:

- Pruning and ordering of rules: Earlier approaches have used greedy methodology to construct the rule set. For example, in CPAR, the feature with maximum FOIL gain is selected at each step to form a rule. This may not be globally optimal because the best feature chosen (in terms of FOIL gain) may be highly correlated with previously chosen features, and therefore does not enhance the classification accuracy. This problem is avoided through Associative classification, in which
Table 7.3 Rules generated for example 1

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Rule</th>
<th>IG</th>
<th>Phase</th>
<th>Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A_j = 1, B_j = 0, C_n = 2 \rightarrow X = 1$</td>
<td>0.372</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>$A_j = 1, B_j = 1 \rightarrow X = 1$</td>
<td>0.5265</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>$B_j = 0, C_n = 1 \rightarrow X = 2$</td>
<td>0.48</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>$A_j = 1, C_n = 2 \rightarrow X = 1$</td>
<td>0.363</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>$A_j = 1, B_j = 0 \rightarrow X = 1$</td>
<td>0.362</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>$A_j = 1, B_j = 0 \rightarrow X = 1$</td>
<td>0.254</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>$A_j = 1, B_j = 0 \rightarrow X = 2$</td>
<td>0.202</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>$A_j = 0 \rightarrow X = 2$</td>
<td>0.479</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>$B_j = 1 \rightarrow X = 1$</td>
<td>0.369</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>$C_n = 1 \rightarrow X = 2$</td>
<td>0.312</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>$C_n = 2 \rightarrow X = 1$</td>
<td>0.21</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>$A_j = 1 \rightarrow X = 1$</td>
<td>0.209</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>$B_j = 0 \rightarrow X = 2$</td>
<td>0.167</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>$B_j = 0 \rightarrow X = 1$</td>
<td>0.157</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>$C_n = 2 \rightarrow X = 2$</td>
<td>0.101</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>$A_j = 1 \rightarrow X = 2$</td>
<td>0.0391</td>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

we mine patterns in the form of association rules by exploring the search space in an exhaustive manner (instead of greedy approach). Though this mining is exhaustive, it is constrained by appropriate minimum support to ensure that the mining is done efficiently. This way the mining picks up most of the dominant and statistically significant patterns and associations between features. The rule set is obtained by considering and comparing each rule with every other rule in the rule-set on the basis of information gain and maximum entropy.

- Classification coverage: Most classifiers rely on a critical phase of feature extraction and pruning which is often guided by experts. Unfortunately, this phase has the potential for missing out on critical features that the expert may not have considered. Such classifiers may miss out on some patterns because some classes may be under-represented or not represented at all. FSEAC ensures that all classes are well represented by obtaining both local and global patterns (rules), and thus optimal in its coverage. Global patterns are mined from the dataset with records pertaining to all classes, and local patterns are mined from records pertaining to each individual class.

The sub-structures of the mining and classifier building process are global association rules and local association rules for each under-represented/non-represented class $y_i \in Y$ where $i = 1$ to $m$. We represent the set $G = G_1 \cup G_2 \cup \ldots \cup G_m$ where $G_i = \{r_{i1}, r_{i2}, \ldots\}; i = 1$ to $m$. $G$ is the super-set containing union of sets, which in turn contain association rules mined globally for each class $y_i \in C$.

For each $G_i = \emptyset$ or $|G_i| < cutOff$, i.e., for every under-represented/non-represented class, we need to mine local association rules. We represent the set $L = L_1 \cup L_2 \cup \ldots \cup L_z$ where $L_j = \{r_{j1}, r_{j2}, \ldots\}; j = 1$ to $z$. $L$ is the super-set containing union of sets, which in turn contain association rules $r_{jl}$ mined locally for each under-represented/non-represented class $y_j \in Y$. Thus, traversing the smallest sub-structures, which are either the sets of global association rules ($G_1, G_2, \ldots, G_m$) or local
association rules \((L_1, L_2, \ldots, L_z)\) for each class \(y_i\), we get sub-structures at the next higher level, \(i.e.\)
\[ G = G_1 \cup G_2 \cup \ldots \cup G_m \quad \text{or} \quad L = L_1 \cup L_2 \cup \ldots \cup L_b, \]
Finally, on further traversing bottom-up and merging \(G\) and \(L\), we get \(F = G \cup L\). \(F\) is the final set of association rules, mined both globally and locally, which is used in the classifier building process to derive classification rules. Thus, this bottom-up traversal of each level of sub-structures provides an accurate solution to the larger problem at hand, \(i.e.\) finding \(F\). This structure (set \(F\)) is achieved in FSEAC through the constrained exhaustive mining of frequent patterns and association rules, and the subsequent bottom-up merging of its constituent sub-structures.

After getting \(F\), we now build the classifier by extracting the set of classification rules \(R\) from \(F\). For each of rule \(r\), we calculate the corresponding information gain \(IG\). This is represented in the form \((r, IG)\). Then, we take each pair of rules \(r_a\) and \(r_b\) with \((r_a, IG_a)\) and \((r_b, IG_b)\), with the rule with higher information gain finding a superior position in the set of ordered classification rules. In the classification process, this rule takes precedence over the other rule which has lower information gain.

Traversing bottom-up, we ultimately order the rule set \(R\) with the best rule at the top. All this is not possible without maintaining a hash table/Hashmap containing the information gain \(IG\) of each rule \(r\) against the rule itself. The final result of all this processing is that we have a sorted rule set \(R\) which is used for the actual classification as described in Section 7.3. And, this rule set is derived from both global and local association rules, thus covering all classes in the given dataset.

### 7.3 Classification Using FSEAC

This section shows how the FSEAC-based classifiers trained in sections 7.2.4.1 and 7.2.4.2 can be used for actual classification. At the conclusion of the second phase of FSEAC, the rule set \(R\) has been pruned in a very efficient and accurate manner. This rule set is obtained after applying ARM both globally (first phase) and locally (second phase). A few more notations are added in this section, in addition to the ones defined previously:

- **ACM** = Hashmap like data structure having consequents \((y \in C)\) of rules \((r \in R)\) which satisfy the unclassified tuple, with the count of how many times rules, with a particular consequent \(y\) satisfy the unclassified tuple.

- **IGM** = Hashmap like data structure having consequents \((y \in C)\) of rules \((r \in R)\) which satisfy the unclassified tuple, with the cumulative information gain of rules, with a particular consequent \(y\) satisfy the unclassified tuple.

- **\(R_{IG'}\)** = Hashmap like data structure having rules (every \(r \in R\)) with their respective fuzzy-membership-weighted information gains \((IGs)\)

An unlabeled tuple \(t'\), containing binary and numeric features, is transformed into a fuzzy tuple, \(t\) containing binary and fuzzy features using a minor variation of the pre-processing technique (FPrep) described in Chapter 3 and [MP10b]. Instead of using a threshold for the fuzzy membership of each fuzzy attribute during fuzzy dataset generation (as is the case in FPrep and in training of FSEAC),
only the partition of each fuzzy attribute having the highest fuzzy membership is used for record/tuple generation during testing/scoring. A tuple with unknown class label is classified using this rule set $R$. The classification (Algorithm 17) of $t$ is a function of minimum (cut off) rule length and best $k\%$ of rules – $f(\text{minlen}, \text{bestk})$.

Each rule set is unique in its characteristics, especially in terms of its size and average rule length. The parameters used for classification can either be user-specified or system defaults. Larger rules are better as they are more specific and have more accuracy [Tha07]. The only possible problem is that they may over-fit. But, this is taken care by ensuring that they satisfy the minimum support specified by the user. And, this fact has been corroborated by our experiments on various datasets, as listed in Section 7.5. We illustrate this fact using a rather simplistic example (though in reality rules are much more complex and longer). Assuming a hypothetical scenario with a vehicle dataset, we may arrive at a rule like “Age = Young $\to$ Vehicle = Suzuki_Hayabusa”. But, the moment we add the attribute-value pair “Gender = Female”, the class may change drastically, the rules becoming “Age = Young, Gender = Female $\to$ Vehicle = Honda_Activa”. In case a test data point has features like “Age = Young”, “Gender = Female”, and a few more, using the former rule would result in an erroneous classification, whereas using the latter would yield the correct result. This small and simplistic example illustrates how the addition of a feature or a few features can drastically change the result of the classification and ultimately affect the accuracy achieved.

For classification, we use a new derived information metric called fuzzy-membership-weighted information gain ($F\text{IG}$), which is given by Eq. 7.4 $\mu$ is the membership of a precedent of a rule in a tuple, and is used to weigh the information gain ($IG$) associated with the rule. $\mu$ can be calculated using a suitable t-norm (Min t-norm in this case). Each rule $r_q$ of $R$ is applied to the unclassified tuple. Only those rules with rule length equal to or greater than minimum rule length ($\text{minlen}$) are used for the classification process (Algorithm 17 line 7). If a rule matches $t$, then the fuzzy-membership-weighted information gain ($F\text{IG}$) is calculated using the minimum fuzzy membership ($\mu$), in $t$, among the items involved in $X$. Calculation of the minimum fuzzy membership is done using the Min t-norm. Then, $r_q$ along with its ($F\text{IG}$) are put in $R_{F\text{IG}}$ (Algorithm 17 line 4–13). The rules in ($F\text{IG}$) are sorted according their fuzzy-membership-weighted information gains (Algorithm 17 line 14).

$$F\text{IG} = \mu \times IG$$

(7.4)

For each rule $r_q$ in $R_{F\text{IG}}$, i.e. rule that matched $t$, the consequent class label $y$ is singled out and the corresponding fuzzy-membership-weighted information gain for the consequent in the rule is accumulated for that particular consequent in $IGM$ (Algorithm 17 line 29). For each consequent, only the top $k\%$ ($\text{bestk}$) of rules is used (Algorithm 17 line 22). After all rules in $R_{F\text{IG}}$ have been evaluated in this manner, the consequent class label with maximum value of $F\text{IG}$ in $IGM_{\text{bestk}}$ is output as the class label for $t$ (Algorithm 17 line 33). Thus, we ascertain the top $k\%$ best rules for each class label $y$, based on fuzzy-membership-weighted information gain, for the given unlabeled tuple $t$, and then to judge the best class label based on the highest cumulative fuzzy-membership-weighted information gain (Algorithm 17 lines 15–32).
7.3.1 Illustration of classification using FSEAC

This section shows the classification process and results for the example classifiers built in sections 7.2.4.1 and 7.2.4.2.

7.3.1.1 Example 1

The sample test tuples (Table 7.4), containing binary and numeric features, are transformed into tuples, containing binary and fuzzy features (Table 7.5) as described in Section 7.3. The tuples shown in Table 7.5 are used for the actual classification process for the classifier built in section 7.2.4.1. The classification is done with best\(k\) = 25\% (translating to the best rule for each consequent class label \(y \in C\)) and \(minlen = 1\). The results are as follows:

- The first tuple is satisfied by rule number 2, pertaining to class \(X = 1\). The values of \(\text{FIG in IGM}_{\text{best}k \times R.\text{size}}\) for \(X = 1\) and \(X = 2\) are 0.254 and 0.000 respectively. Thus, \(X = 1\) is predicted as the class label.
- For the second tuple, the values of \(\text{FIG in IGM}_{\text{best}k \times R.\text{size}}\) for \(X = 1\) and \(X = 2\) are 0.000 and 0.471 respectively. Thus, \(X = 2\) is predicted as the class label. Rule numbers 3 and 8 satisfy this tuple.
- Last, for the third tuple, the values of \(\text{FIG in IGM}_{\text{best}k \times R.\text{size}}\) for \(X = 1\) and \(X = 2\) are 0.18 and 0.000 respectively. Thus, \(X = 1\) is predicated as the class label. Only rule number 1 satisfies this tuple.

| Table 7.4 Sample test dataset for example 1 |
|------------------|------------------|------------------|------------------|
| \(A_n\) | \(B_n\) | \(C_b\) | \(X\) |
| 1 | 1 | 1 | 1 |
| 4 | 4 | 1 | 2 |
| 1 | 3.8 | 2 | 1 |

| Table 7.5 Sample test dataset for example 1 |
|------------------|------------------|------------------|------------------|
| \(A_f\) | \(B_f\) | \(C_b\) | \(X\) |
| 1\text{\textcircled{}}0.9668686 | 1\text{\textcircled{}}0.96894675 | 1\text{\textcircled{}}1 | 1 |
| 0\text{\textcircled{}}0.9951109 | 0\text{\textcircled{}}0.9707756 | 1\text{\textcircled{}}1 | 2 |
| 1\text{\textcircled{}}0.9668686 | 0\text{\textcircled{}}0.9764175 | 2\text{\textcircled{}}1 | 1 |

7.3.1.2 Example 2

As mentioned in section 7.2.4.2, we have used 10\% of the hepatic dataset for testing. We use one such unclassified test tuple to illustrate how classification can be done using FSEAC. We use best\(k\) = 5\% and \(minlen = 4\). The size of \(R\) is 510 rules, so in absolute terms best\(k\) = 12\% would translate to 60 best rules for each consequent class label \(y \in C\).

Taking one sample instance of a tuple being classified, we see that for the parameters specified in the paragraph above, rules with both consequents, \(Class = 1\) and \(Class = 2\), satisfy the tuple. 1 rule with
class label \( \text{Class} = 1 \) and 60 rules with class label \( \text{Class} = 2 \) satisfy this tuple. The class label with maximum value of \( \text{FIG in IGM} \) is output as the class label for any tuple. In this case \( \text{Class} = 1 \) has a value of 0.0004 and \( \text{Class} = 2 \) has a value of 0.0788. Thus, \( \text{Class} = 2 \) is output as the class label for this particular test tuple.

### 7.3.1.3 Example 3

This example is to illustrate how multiple rules are used to achieve better accuracy. Using a single rule or very few rules may result in improper output, as the classification is not supported enough by the rule-base. The same problem can occur if we use a large number of rules, which in effect amounts to over-fitting. Thus, we need to use a small number of optimal rules to get the correct classification. This small number of rules is specified by \( \text{bestk} \) best rules, with \( k \) varying according to the dataset and domain. Given a test data point “Age = Young, Gender = Male, Income = Very_High, Vehicle = Suzuki_Hayabusa”, let us assume that the following three rules form a part of the rule-base:

- \( R_1 \): “Gender = Male, Income = Very_High \rightarrow Vehicle = Honda_Accord”
- \( R_2 \): “Age = Young, Gender = Male \rightarrow Vehicle = Suzuki_Hayabusa”
- \( R_3 \): “Age = Young, Income = Very_High \rightarrow Vehicle = Suzuki_Hayabusa”

Rules would be sorted on rule length first and then the information gain, as shown above. In this case, without loss of generality, we assume all the three rules have the same information gain. If \( \text{bestk} = 33\% \), i.e. only one rule for each class would be used for classification. But this would result in a wrong classification as \( R_1 \) and \( R_2 \) would be applied, with both rules having different class labels and neither of them winning outright. On the other hand, with \( \text{bestk} = 66\% \), i.e. up to two rules per class being used, we get the correct classification (“Vehicle = Suzuki_Hayabusa”). \( R_2 \) and \( R_3 \) are applied for “Vehicle = Suzuki_Hayabusa” whereas only \( R_1 \) is applied for “Vehicle = Honda_Activa”.

### 7.4 Parameter configuration in FSEAC

Apart from its accuracy, the other salient feature of FSEAC is its ease of use. The parameters which are involved in FSEAC are easy to configure. The parameters discussed in this section are:

- \( \text{cutoff} \) – threshold to check for under-represented classes which would need local ARM in second phase (training)
- \( \text{minlen} \) – threshold for minimum rule length for classification (testing)
- \( \text{bestk} \) – best \( k\% \) rules used for classification (testing)

To illustrate the configuration of these parameters, we take an example (example 4) of a dataset with five classes, namely \( \text{Class} = 1 \), \( \text{Class} = 2 \), \( \text{Class} = 3 \), \( \text{Class} = 4 \), and \( \text{Class} = 5 \). In the first phase of FSEAC we get classification rules for each class, as shown in Table 7.6. The size of the rule set...
at this point is 50 rules, with \( \text{Class} = 5 \) having no classification rule at all, and \( \text{Class} = 1 \) having the highest number of rules, i.e. 25. Assuming \( \text{cutoff} = 60\% \), for any class to be adequately represented in the rule set, it should at least have \( 60\% \times 25 \) (highest number of rules for any class) = 15 rules. Consequently, local ARM is carried out for \( \text{Class} = 3 \) and \( \text{Class} = 4 \) (< 20 rules in the first phase), along with \( \text{Class} = 5 \) (no rules at all in the first phase). After local ARM in the second phase, the size of the rule set grows to 90 rules, with the under-represented/non-represented classes now having adequate representation. Subsequently, all rules in \( R \) are sorted on two criteria – rule length (number of items/literals on the left-hand-side/precedent of the rule) being the primary criterion and information gain of each rule the secondary criterion.

This classifier is then used for the actual classification process, which in turn uses two relative parameters. The maximum rule length in this case is seven, so with \( \text{minlen} = 4 \), which means only a rule with at least four items in its precedent would be used for classification. And, \( \text{bestk} = 5\% \) translates to \( 5\% \times 90 \) (rule set size) = 4.5 \( \approx \) 5 rules. Thus, the best five rules for each consequent class label \( y \in C \), are used for the actual classification process. For each class the top five rules, with rule length not less than four items, are applied to get the information gain. Based on the information gain for each class, the class with the best information gain wins (please refer section 7.3.1 for more details).

<table>
<thead>
<tr>
<th>Class</th>
<th>Phase 1</th>
<th>Phase 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>–</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>–</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>All classes</td>
<td>50</td>
<td>40</td>
<td>90</td>
</tr>
</tbody>
</table>

### 7.5 Performance Study

In this section, we describe the performance study model that has been used for testing FSEAC using 23 disparate UCI Machine Learning (ML) datasets. The characteristics of the datasets are described in Table 7.7. For crisp classifiers numerical attributes in all datasets have been converted to binary attributes using sharp bucket-like partitions, and for fuzzy classifiers numerical attributes have been converted to fuzzy attributes. The data range and size of each partition are determined by the values (their respective frequencies) for each numerical attribute in the dataset. The performance of FSEAC has been compared with that of other popular contemporary state-of-the-art classifiers, and the results of the same are detailed in Section 7.5.1. Specifically, the classifiers used for comparisons are:

- Simple and Effective Associative Classifier (SEAC) – Chapter 6
- Classification based on Predictive Association Rules (CPAR) [YH03]
- Classification based on Multiple Rules (CMAR) [LHP01]
• Classification based on Associations (CBA) [LHM98]
• Associative Classifier based on Maximum Entropy Principle (ACME) [TP05]
• C4.5 [Qui93]
• Ripper [Coh95]
• Tree-Augmented naive-bayes (TAN) [FGG97]
• Naive-bayes (NB) [FGG97]
• Lazy Associative Classification - LAC (information gain) [VJJ06]
• Lazy Associative Classification - LAC (confidence) [VJJ06]
• Lazy Associative Classification - L^3 (tuned) [BCG08]
• Support Vector Machines (SVM) – Linear [DBK96]
• An algorithm for unordered fuzzy rule induction (FURIA) [HH09b]
• A Fuzzy Rule Learner for Inducing Reliable Classifiers (FR3) [HH09a]
• Round Robin Classification (R3) [Fur02]
• A genetic learning system based on an iterative approach (SLAVE) [GP99]

In all the experiments (of FSEAC and other associative classification algorithms), accuracy is measured using the 10-fold cross validation. We have implemented FSEAC using Java on an Intel i7 PC with 4 GB main memory. The results of the remaining algorithms have been taken from [TP05], [YH03], [VJJ06], [BCG08], [HH09b], and [HH09a].

For FSEAC, the configuration of various parameters for fuzzy pre-processing, fuzzy ARM, and classifier training and scoring is detailed below:

• num_clusters – The number of clusters used for each numerical attribute for fuzzy c-means clustering during fuzzy pre-processing. For all the datasets, num_clusters took the values 2, 3, or 4 (Table 7.8).

• minsup – The same minimum support is used for both global and local ARM in any dataset. minsup needs to be fixed according to the characteristics of the dataset, like density, dataset size and number and type of features. For the current experimental study minsup was either 5% or 30% for the datasets used (Table 7.8).

• minconf – The same minimum confidence value of 85% has been used for all the datasets.
### Table 7.7 Characteristics of Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Records</th>
<th>Numerical</th>
<th>Binary</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>austral</td>
<td>690</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>anneal</td>
<td>898</td>
<td>6</td>
<td>33</td>
<td>6</td>
</tr>
<tr>
<td>breast</td>
<td>699</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>cleve</td>
<td>303</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>crx</td>
<td>690</td>
<td>6</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>diabetes</td>
<td>768</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>german</td>
<td>1000</td>
<td>7</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>9</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>heart</td>
<td>270</td>
<td>6</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>hepatic</td>
<td>155</td>
<td>6</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>horse</td>
<td>36</td>
<td>7</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>hypo</td>
<td>3163</td>
<td>7</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>iono</td>
<td>351</td>
<td>32</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>labor</td>
<td>57</td>
<td>7</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>led7</td>
<td>3200</td>
<td>0</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>lymph</td>
<td>148</td>
<td>3</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>pima</td>
<td>768</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>sick</td>
<td>2800</td>
<td>7</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>sonar</td>
<td>208</td>
<td>60</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>tic-tac</td>
<td>958</td>
<td>0</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>waveform</td>
<td>5000</td>
<td>21</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>wine</td>
<td>178</td>
<td>13</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

• *cutoff* – The threshold to check if local ARM should be performed for a particular class in the second phase of FSEAC. *cutoff* can be used to provide more coverage to under-represented classes. The default value of 0.5 has been used for two datasets. For the rest of the datasets, *cutoff* > 1 (any value > 1) has been used, so that local ARM is performed for all classes in the second phase (Table 7.8).

• *minlen* – This is the threshold for the minimum rule length used for scoring during classification. *minlen* took the values 2, 3, or 4 for all the datasets used (Table 7.8).

• *bestk* – The best k% rules used for scoring during classification. The value of *bestk* for the datasets used for experimentation was either 5% or 10% (Table 7.8).

*minsup* and *minconf* are normal minimum support and minimum confidence used in ARM. They need to be configured in the same manner as we configure minimum support and minimum confidence for any ARM algorithm. The minimum support and minimum confidence values rely on how dense or sparse the dataset is, the number of items (singletons) involved in the dataset, and the average length of transactions in the dataset [ZKM01]. Moreover, once a rule set has been generated for a given dataset,
one can look at the rules and the rule set size to decide the parameters $best_k$ and $minlen$. Thus, deciding these two parameters, which are specific to FSEAC, is easy and straightforward.

### Table 7.8 Configurations of parameters for Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>num_clusters</th>
<th>minsup</th>
<th>cutoff</th>
<th>minlen</th>
<th>bestk</th>
</tr>
</thead>
<tbody>
<tr>
<td>austral</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>10%</td>
</tr>
<tr>
<td>anneal</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>10%</td>
</tr>
<tr>
<td>breast</td>
<td>3</td>
<td>5%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>cleve</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>crx</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>diabetes</td>
<td>3</td>
<td>5%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>german</td>
<td>2</td>
<td>30%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>glass</td>
<td>3</td>
<td>5%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>heart</td>
<td>2</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>hepatic</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>horse</td>
<td>2</td>
<td>30%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>hypo</td>
<td>2</td>
<td>5%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>iono</td>
<td>3</td>
<td>30%</td>
<td>0.5</td>
<td>2</td>
<td>10%</td>
</tr>
<tr>
<td>iris</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>4</td>
<td>10%</td>
</tr>
<tr>
<td>labor</td>
<td>2</td>
<td>5%</td>
<td>&gt; 1</td>
<td>2</td>
<td>10%</td>
</tr>
<tr>
<td>led7</td>
<td>3</td>
<td>5%</td>
<td>&gt; 1</td>
<td>2</td>
<td>10%</td>
</tr>
<tr>
<td>lymph</td>
<td>3</td>
<td>5%</td>
<td>0.5</td>
<td>4</td>
<td>10%</td>
</tr>
<tr>
<td>pima</td>
<td>4</td>
<td>5%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>sick</td>
<td>3</td>
<td>5%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>sonar</td>
<td>3</td>
<td>30%</td>
<td>0.5</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>tic-tac</td>
<td>–</td>
<td>5%</td>
<td>&gt; 1</td>
<td>4</td>
<td>5%</td>
</tr>
<tr>
<td>waveform</td>
<td>3</td>
<td>30%</td>
<td>&gt; 1</td>
<td>2</td>
<td>5%</td>
</tr>
<tr>
<td>wine</td>
<td>2</td>
<td>30%</td>
<td>&gt; 1</td>
<td>2</td>
<td>10%</td>
</tr>
</tbody>
</table>

### 7.5.1 Experimental results

As mentioned above, 23 UCL-ML datasets have been used to illustrate the efficacy of FSEAC in terms of accuracy and simplicity of use. Table 7.9 illustrates the experimental results to test accuracy obtained for various classifiers on each of the 23 UCI-ML datasets. Of the 23 datasets, FSEAC has the best accuracy for 17 of them as compared to other classification algorithms, prominent of which are SEAC, L$^3$, LAC, SVM, CPAR, CMAR, FR3, FURIA, and ACME. For three other datasets, FSEAC ranks in the top three among all 16 classifiers compared. Of the 17 datasets for which FSEAC has the best accuracy, there are 12 datasets (austral, cleve, crx, glass, heart, iris, labor, led7, lymph, sonar, waveform, and wine) such that for each dataset the accuracy of FSEAC and that of the second best algorithm (except SEAC) differs by an absolute margin of 0.5% or more. FSEAC also performs better than SEAC, thus indicating that Fuzzy Logic helps in achieving better accuracy, especially in real-life datasets which generally have a mix of numerical and binary attributes. We can also see that FSEAC performs very well against all kinds of datasets as compared to a host of classifiers, each of which is based on a different principle and uses different techniques. This highlights the efficacy of Fuzzy
Associative Classifiers, more specifically FSEAC. In addition to this, FSEAC is very transparent, i.e. we can see the rules, which makes the configuration of various parameters intuitive and very easy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FSE-AC</th>
<th>SE-AC</th>
<th>CP-AR</th>
<th>CM-AR</th>
<th>CBA</th>
<th>AC-ME</th>
<th>C4.5</th>
<th>TAN</th>
<th>Ripper</th>
<th>NB</th>
<th>LAC (inf. gain)</th>
<th>LAC (co-inf.)</th>
<th>SVM (Linear)</th>
<th>L^2-tunned</th>
<th>FR3</th>
<th>R3</th>
<th>FU-RIA</th>
<th>SL-AVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>austera</td>
<td>87.9%</td>
<td>87.8</td>
<td>86.2</td>
<td>86.1</td>
<td>84.9</td>
<td>85.36</td>
<td>84.7</td>
<td>84.9</td>
<td>87.3</td>
<td>84.34</td>
<td>87.3</td>
<td>86.6</td>
<td>85.5</td>
<td>87</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>annual</td>
<td>95.5</td>
<td>–</td>
<td>98.4</td>
<td>97.3</td>
<td>97.9</td>
<td>–</td>
<td>94.5</td>
<td>–</td>
<td>95.8</td>
<td>–</td>
<td>96.5</td>
<td>96.6</td>
<td>96.8</td>
<td>98.1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>breast</td>
<td>97.5%</td>
<td>97.3</td>
<td>96</td>
<td>96.4</td>
<td>96.3</td>
<td>96.49</td>
<td>95</td>
<td>96.5</td>
<td>95.1</td>
<td>97.28</td>
<td>96.8</td>
<td>95.9</td>
<td>97.6</td>
<td>96.4</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>cleave</td>
<td>86.9%</td>
<td>86.6</td>
<td>81.5</td>
<td>82.2</td>
<td>82.8</td>
<td>83.82</td>
<td>78.2</td>
<td>82.5</td>
<td>82.2</td>
<td>83.82</td>
<td>84.4</td>
<td>83.3</td>
<td>83.8</td>
<td>85.5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>crx</td>
<td>87.3%</td>
<td>–</td>
<td>85.7</td>
<td>84.9</td>
<td>84.7</td>
<td>–</td>
<td>84.9</td>
<td>–</td>
<td>84.9</td>
<td>–</td>
<td>85.8</td>
<td>85.9</td>
<td>85.5</td>
<td>86.7</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>diabetes</td>
<td>77.3</td>
<td>76.7</td>
<td>75.1</td>
<td>75.8</td>
<td>74.5</td>
<td>77.86</td>
<td>74.2</td>
<td>76.3</td>
<td>74.7</td>
<td>75.78</td>
<td>80.3</td>
<td>78.7</td>
<td>76.7</td>
<td>78.7</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>german</td>
<td>77.4%</td>
<td>74.7</td>
<td>73.4</td>
<td>74.9</td>
<td>73.4</td>
<td>71.3</td>
<td>72.3</td>
<td>73</td>
<td>69.8</td>
<td>70</td>
<td>76.6</td>
<td>78</td>
<td>76.2</td>
<td>74.3</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>glass</td>
<td>80.9%</td>
<td>77.4</td>
<td>74.4</td>
<td>70.1</td>
<td>73.9</td>
<td>–</td>
<td>68.7</td>
<td>–</td>
<td>69.1</td>
<td>–</td>
<td>74</td>
<td>72.6</td>
<td>60.8</td>
<td>78</td>
<td>72.98</td>
<td>69.61</td>
<td>68.22</td>
<td>61.83</td>
</tr>
<tr>
<td>heart</td>
<td>86%</td>
<td>85.7</td>
<td>82.6</td>
<td>82.2</td>
<td>81.9</td>
<td>82.96</td>
<td>80.8</td>
<td>81.8</td>
<td>80.7</td>
<td>83.7</td>
<td>83.1</td>
<td>82.8</td>
<td>82.3</td>
<td>84.4</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>hepatic</td>
<td>89.4%</td>
<td>85.2</td>
<td>79.4</td>
<td>80.5</td>
<td>81.8</td>
<td>82.58</td>
<td>80.6</td>
<td>81.29</td>
<td>76.7</td>
<td>80.22</td>
<td>82.9</td>
<td>84.9</td>
<td>89</td>
<td>85.8</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>horse</td>
<td>85.9%</td>
<td>–</td>
<td>84.2</td>
<td>82.6</td>
<td>82.1</td>
<td>–</td>
<td>82.6</td>
<td>–</td>
<td>84.8</td>
<td>–</td>
<td>85.5</td>
<td>82.8</td>
<td>85.1</td>
<td>84.2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>hypo</td>
<td>99.3%</td>
<td>–</td>
<td>98.1</td>
<td>98.4</td>
<td>98.9</td>
<td>–</td>
<td>99.2</td>
<td>–</td>
<td>98.9</td>
<td>–</td>
<td>99</td>
<td>98.8</td>
<td>99.2</td>
<td>98.9</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>iono</td>
<td>93.7%</td>
<td>–</td>
<td>92.6</td>
<td>91.5</td>
<td>92.3</td>
<td>–</td>
<td>90</td>
<td>–</td>
<td>91.2</td>
<td>–</td>
<td>92.2</td>
<td>91.5</td>
<td>92</td>
<td>93.5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>89.59</td>
</tr>
<tr>
<td>iris</td>
<td>97.4%</td>
<td>96.9</td>
<td>94.7</td>
<td>94</td>
<td>94.7</td>
<td>–</td>
<td>95.3</td>
<td>–</td>
<td>94</td>
<td>–</td>
<td>96.8</td>
<td>95.4</td>
<td>94.7</td>
<td>93.3</td>
<td>94.78</td>
<td>94.25</td>
<td>94.76</td>
<td>94.92</td>
</tr>
<tr>
<td>labor</td>
<td>95.7%</td>
<td>–</td>
<td>84.7</td>
<td>89.7</td>
<td>86.3</td>
<td>–</td>
<td>79.3</td>
<td>–</td>
<td>84</td>
<td>–</td>
<td>80.9</td>
<td>83</td>
<td>89.5</td>
<td>94.7</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>led7</td>
<td>79.2%</td>
<td>78.5</td>
<td>73.6</td>
<td>72.5</td>
<td>71.9</td>
<td>–</td>
<td>73.5</td>
<td>–</td>
<td>69.7</td>
<td>–</td>
<td>77.9</td>
<td>74.5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>lymph</td>
<td>87.7%</td>
<td>79.9</td>
<td>82.3</td>
<td>83.1</td>
<td>77.8</td>
<td>78.4</td>
<td>73.5</td>
<td>84.45</td>
<td>79</td>
<td>83.1</td>
<td>80.9</td>
<td>81.8</td>
<td>77.7</td>
<td>86.5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>pima</td>
<td>78%</td>
<td>77.6</td>
<td>73.8</td>
<td>75.1</td>
<td>72.9</td>
<td>77.89</td>
<td>75.5</td>
<td>76.3</td>
<td>73.1</td>
<td>76.17</td>
<td>78</td>
<td>78.7</td>
<td>76.6</td>
<td>78.5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>74.71</td>
</tr>
<tr>
<td>sick</td>
<td>96.9%</td>
<td>–</td>
<td>96.8</td>
<td>97.5</td>
<td>97</td>
<td>–</td>
<td>98.5</td>
<td>–</td>
<td>97.7</td>
<td>–</td>
<td>98</td>
<td>98</td>
<td>96.8</td>
<td>95.6</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>sonar</td>
<td>90.9%</td>
<td>–</td>
<td>79.3</td>
<td>79.4</td>
<td>77.5</td>
<td>–</td>
<td>70.2</td>
<td>–</td>
<td>78.4</td>
<td>–</td>
<td>79.5</td>
<td>80.4</td>
<td>84.6</td>
<td>81.7</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>77.01</td>
</tr>
<tr>
<td>tic-tac</td>
<td>99.8%</td>
<td>99.8</td>
<td>98.6</td>
<td>99.2</td>
<td>99.6</td>
<td>–</td>
<td>99.4</td>
<td>–</td>
<td>98</td>
<td>–</td>
<td>99.4</td>
<td>100</td>
<td>98.3</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>wave</td>
<td>85.2%</td>
<td>80.7</td>
<td>80.9</td>
<td>83.2</td>
<td>80</td>
<td>83.08</td>
<td>78.1</td>
<td>81.52</td>
<td>76</td>
<td>–</td>
<td>80.7</td>
<td>81.1</td>
<td>84.3</td>
<td>82.5</td>
<td>79.79</td>
<td>80.58</td>
<td>82.84</td>
<td>75.34</td>
</tr>
<tr>
<td>wine</td>
<td>98.9%</td>
<td>98.8</td>
<td>95.5</td>
<td>95</td>
<td>95</td>
<td>–</td>
<td>92.7</td>
<td>–</td>
<td>91.6</td>
<td>–</td>
<td>96.6</td>
<td>96.1</td>
<td>98.3</td>
<td>92.7</td>
<td>92.63</td>
<td>93.25</td>
<td>92.46</td>
<td></td>
</tr>
</tbody>
</table>

# – best for that particular dataset

* – within top 3 for that particular dataset

### 7.5.2 FSEAC and its efficiency and accuracy

FSEAC is a very effective and accurate algorithm, but at the same time is very easy-to-use. These can be attributed to the following features of FSEAC:

- It does not use a greedy strategy (like CMAR and CPAR), but on the contrary uses a constrained exhaustive approach.
• Use of Fuzzy Logic helps in the soft partitioning of numerical attributes. This eliminates uncertainty, especially at partition boundaries. This uncertainty is found when sharp partitioning is used.

• Fuzzy Logic also helps avoid polysemy and synonymy. The fuzzy partitions are mathematically and automatically (unlike manually in case of sharp partitions) determined using Fuzzy c-means clustering.

• FSEAC deals with redundant association rules through simple and effective pruning in order to reduce the number of rules.

• Training is done in a two-phased manner in order to obtain maximum representation of all possible classes involved.

• Classifier training is done with minimum user intervention using easy-to-configure parameters, like best $k\%$ rules and minimum rule length.

• Rule pruning, sorting, analysis, and ultimately classification is based on entropy and information gain, universally accepted as some of the best measures of information.

• Like SEAC, it does not use ancillary techniques like FOIL, PRM and GIS (as opposed to ACME and CPAR), as they add an overhead to classifier training in terms of performance and accuracy.

• Only the lazy associative classifiers, $L^3$ and LAC, have accuracies comparable to that of FSEAC. This is primarily because of the exhaustive manner in which they process rules.

Thus, it is very clear from the above that FSEAC has been formulated using a totally different perspective as compared to contemporary associative classification algorithms.

### 7.6 Summary of FSEAC

We have presented a novel associative classifier called FSEAC which is very straightforward and easy to use, and also very accurate in the classification results that it produces. FSEAC uses a constrained exhaustive approach which is pivotal in its performance as far as accuracy is concerned, and does not use any greedy approach. Its accuracy is clearly highlighted in section 7.5.1 where it has been compared to the best contemporary classification algorithms, associative as well as non-associative ones.

FSEAC uses an effective and simple pruning technique based on a new metric which we have come up with called Fuzzy-weighted Information gain. Largely because of the fuzzy flavour, FSEAC provides better accuracy as compared to SEAC, as is evident from our experimental studies. This is because of the ability of FSEAC in dealing with numerical attributes in an effective manner through soft partitions.
\begin{algorithm}
\caption{Pseudo-code for phase 1}

1: $CM = \emptyset$
2: $RCM = \emptyset$
3: $R_{IG} = \emptyset$
4: $S = \emptyset$
5: $R = \text{far\_miner}(E, \text{minsup}, \text{minconf})$
6: \textbf{for} each $r_q \in R$ where $q = 1$ to $n$ \textbf{do}
7: \hspace{1em} /* $r_q$ is of the form $X \rightarrow Y$ */
8: \hspace{1em} $IG = H(Y) - H(Y|X)$
9: \hspace{1em} put $(r_q, IG)$ in $R_{IG}$
10: \textbf{end for}
11: /* pruning step - start */
12: prune()
13: /* pruning step - end */
14: \textbf{for} each $r_q \in R$ where $q = 1$ to $n$ \textbf{do}
15: \hspace{1em} /* $r_q$ is of the form $X \rightarrow Y$ */
16: \hspace{1em} $y =$ class label which forms the consequent $Y$ of $r_q$
17: \hspace{1em} /* increment count of $y$ in $RCM$ by 1 */
18: \hspace{1em} if $y \in RCM$ then
19: \hspace{2em} $e =$ current count of $y$ in $RCM$
20: \hspace{2em} $e += 1$
21: \hspace{1em} \textbf{else}
22: \hspace{2em} $e = 1$
23: \hspace{1em} \textbf{end if}
24: \hspace{1em} update $(y, e)$ in $RCM$
25: \textbf{end for}
26: $g_{\text{max}} =$ max count for all $ys$ in $RCM$
27: \textbf{for} each $y_h \in C$ where $h = 1$ to $m$ \textbf{do}
28: \hspace{1em} $g =$ count of $y_h$ from $RCM$
29: \hspace{1em} \textbf{if} $y_h \notin RCM$ or $g < \text{cutOff} \times g_{\text{max}}$ \textbf{then}
30: \hspace{2em} /* $y_h$ should be processed locally in phase 2 */
31: \hspace{2em} $S = S \cup y_h$
32: \hspace{1em} \textbf{end if}
33: \textbf{end for}
\end{algorithm}
1: prune()
2: for each \( r_q \in R \) where \( q = 1 \) to \((n - 1)\) do
3:   \{/* \( r_q \) is of the form \( X \rightarrow Y \)*\}
4:   \( y = \) class label which forms the consequent \( Y \) of \( r_q \)
5:   \( IG = \) info gain of \( r_q \) from \( R_{IG} \)
6: for each \( r_s \in R \) where \( s = (q + 1) \) to \( n \) do
7:   \{/* \( r_s \) is of the form \( X' \rightarrow Y' \)*\}
8:   \( y' = \) class label which forms the consequent \( Y' \) of \( r_s \)
9:   \( IG' = \) info gain of \( r_s \) from \( R_{IG} \)
10: if \( y \neq y' \) or \( IG = IG' \) then
11:   if \( r_q \subset r_s \) then
12:     \{/* remove \( r_q \) from \( R \)*\}
13:     \( R = R - r_q \)
14:     \{/* remove \( r_q \) and its info gain from \( R_{IG} \)*\}
15:     \( R_{IG} = R_{IG} - r_q \)
16:     break from loop
17: else if \( r_s \subset r_q \) then
18:     \{/* remove \( r_s \) from \( R \)*\}
19:     \( R = R - r_s \)
20:     \{/* remove \( r_s \) and its info gain from \( R_{IG} \)*\}
21:     \( R_{IG} = R_{IG} - r_s \)
22: end if
23: end if
24: end for
25: end for

Algorithm 15: Pseudo-code for function prune()
1: $R' = \emptyset$
2: $PCM = \emptyset$
3: for each $y \in C$ do
4:  {/* apply Apriori locally for each $y$ */}
5:  $R'' = \text{far\_miner}(E, \text{minsup}, y)$
6:  $R' = R' \cup R''$
7: end for
8: for each $t \in E$ do
9:  for each $r \in R'$ do
10:  {/* $r$ is of the form $X \rightarrow Y$ */}
11:  $y = \text{class label which forms the consequent } Y \text{ of } r$
12:  $X = \text{precedent of } r$
13:  if $X \subseteq t$ then
14:  {/* Min t-norm used */}
15:  $\mu = \text{minimum fuzzy membership, in } t, \text{ of items in } X$
16:  if $X \in PCM$ then
17:  $b = \text{current count of } X \text{ in } PCM$
18:  $b += \mu$
19:  else
20:  $b = \mu$
21:  end if
22: end if
23: end for
24: end for
25: for each $r \in R'$ do
26:  {/* $r$ is of the form $X \rightarrow Y$ */}
27:  $y = \text{class label which forms the consequent } Y \text{ of } r$
28:  $IG = H(Y) - H(Y \mid X)$
29:  $R = R \cup r$
30:  put $(r, IG)$ in $R_{IG}$
31: end for
32: prune()

Algorithm 16: Pseudo-code for phase 2
1: $t =$ input tuple to be classified
2: $ACM = \emptyset$
3: $IGM = \emptyset$
4: for each $r_q \in R$ where $q = 1$ to $n$ do
5:  {/* $r_q$ is of the form $X \rightarrow Y$ */}
6:  $IG' = $ info gain of $r_q$ in $R_{IG}$
7:  if $X \subseteq t$ and $X.length < minlen$ then
8:  {/* Min t-norm used */}
9:  $\mu =$ minimum fuzzy membership, in $t$, of items in $X$
10:  $FIG = \mu \times IG'$
11:  put $(r_q, FIG)$ in $R_{FIG}$
12: end if
13: end for
14: sort all rules in $R_{FIG}$ according to fuzzy-membership-weighted info gain
15: for each $r_q \in R_{FIG}$ where $q = 1$ to $n$ do
16:  {/* $r_q$ is of the form $X \rightarrow Y$ */}
17:  $FIG' = \mu$-weighted info gain of $r_q$ in $R_{FIG}$
18:  if $y \in ACM$ then
19:  $c =$ current count of $y$ in $ACM$
20:  $FIG =$ current info gain of $y$ in $IGM$
21: end if
22: if $y \not\in ACM$ or $c \leq bestk \times R.size$ then
23:  if $y \in ACM$ then
24:  $c += 1$
25: else
26:  $c = 1$
27: end if
28: update $(y, c)$ in $ACM$
29: $FIG^+ = FIG'$
30: update $(y, FIG)$ in $IGM$
31: end if
32: end for
33: print $y$ with max value of $FIG$ in $IGM_{bestk \times R.size}$ as class label for $t$

Algorithm 17: Pseudo-code for classification