Chapter 8

FACISME: Fuzzy Associative Classification using Iterative Scaling and Maximum Entropy

The FACISME \cite{MP10a} algorithm integrates maximum-entropy-based associative classification with fuzzy logic. FACISME is a fuzzy associative classification algorithm which uses maximum entropy and iterative scaling to build a theoretically sound classifier which is meant for accurate and efficient performance on any kind of datasets (irrespective of size and type of attributes – numerical or binary). The maximum entropy principle is well-accepted in the statistics community. Using maximum entropy, given a collection of known facts about a probability distribution, we choose a model for this distribution that is consistent with all the facts, but otherwise is as uniform as possible. Hence, the chosen model for FACISME does not assume any independence among its parameters that is not reflected in the given facts. FACISME uses the Generalized Iterative Scaling (GIS) algorithm \cite{DR72} to compute the maximum entropy model. Because of the use of maximum entropy, FACISME has a very strong theoretical foundation, and does not assume independence of parameters in the classification process, thus providing very good accuracy. Moreover, maximum entropy models are interesting because of their ability to combine many different kinds of information. This accuracy is easily extensible over any kind of datasets (irrespective of size and type of attributes – numerical or binary) and domains, through the use of fuzzy logic, by creating a fuzzy associative classifier.

8.1 Maximum Entropy Models

Let $K = k_1, k_2, \ldots, k_n$ be the set of all items that can appear in a transaction. Conditional maximum entropy models (the term “conditional” is generally dropped in normal usage) are of the form:

$$P(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{\sum_y e^{\sum_i \lambda_i f_i(x,y')}}$$  \hspace{1cm} (8.1)

where $X = x_1, x_2, \ldots, x_{2^n-1}$ is an input vector representing the set of all possible transactions with each transaction containing one or more items from $K$. And, $y$ is an output class such that $y \in Y = y_1, y_2, \ldots, y_l$ – the set of all $l$ classes. $f_i$ is the indicator function or feature value that is true if a particular property of $(x, y)$ is true, and $\lambda_i$ is a weight for the indicator $f_i$. Let $S = \{s_1, s_2, \ldots, s_{|S|}\}$ be the union of the fuzzy frequent itemsets extracted from each class $y$. 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 described in Chapter 3. Maximum entropy models have several valuable properties, of
which the most important is constraint satisfaction. For a given $f_i$, we can count how many times $f_i$ was
observed in the data:

$$
observed[i] = \sum_j f_i(x_j, y_j)
$$

(8.2)

For a model $P_{\lambda'}$ with parameters $\lambda'$, we can see how many times the model predicts that $f_i$
would be expected, so:

$$
expected[i] = \sum_{j,y} P_{\lambda'}(y|x_j) f_i(x_j, y)
$$

(8.3)

The itemsets in $S$ are parameters to model each class, such that each fuzzy itemset $s$ together with
$expected[i]|y$ in class $y$, forms a constraint that needs to be satisfied by the statistical model for that
particular class. Thus, for each class $y$ we have a set of constraints $C[y] = s_q|s_q \in S$. The probability
distribution that we build for class $y$ must satisfy these constraints. However, there could be multiple
distributions satisfying these constraints. But, we use the maximum entropy principle to select the
distribution with the highest entropy. Maximum entropy models have the property that $expected[i] =$
$observed[i]$ for all constraints $i$. Moreover, among models in the form of Eq. 8.1 the maximum entropy
model maximizes the probability of the training data. Yet another property is that maximum entropy
models are as close as possible to the uniform distribution, subject to constraint satisfaction.

### 8.2 FACISME And Fuzzy Associative Classification

In this section we describe how FACISME is used to build a fuzzy associative classifier, and how
this classifier is used for actual classification. Before the classifier training can ensue, we need to extract
the fuzzy frequent itemsets from fuzzy dataset $E$. This can be done using any popular fuzzy ARM
algorithm, with most of such algorithms being fuzzy adaptations of Apriori. But, in this case, we use
the fuzzy ARM algorithm that we have developed in [MP09] (and in Chapter 4). The dataset also
needs to go through appropriate pre-processing before fuzzy association rules can be mined. We use the
pre-processing methodology described in Chapter 3 for this purpose.

In FACISME, the training phase involves finding the set of constraints $i.e. S$ (frequent itemsets
extracted using fuzzy ARM), and computing $\lambda$ values for all the classes. These $\lambda$ values indicate the
weights of the fuzzy frequent itemsets in each class $y$. The computed $\lambda$ values for each class are stored,
and are used in the actual classification phase. We use the classical GIS algorithm to deal with maximum
entropy models. In each iteration of GIS, a step is taken in a direction that increases the likelihood of the
training data, with the step size being neither too large nor too small. The likelihood of the training data
increases at each iteration and eventually converges to the global optimum. GIS converges such that, for
each $f_i$, $expected[i] = observed[i]$. Whenever they are not equal, we can move them closer. To avoid
very small probability and likelihood values, GIS is generally used in its logarithmic form. In this form,
we add \(\log(\text{observed}[i]/\text{expected}[i])\) to \(\lambda_i\), but along with a slowing factor, \(f\#\) (Eq. 8.4), equal to the largest total value of:

\[
f\# = \max_{j,y} \sum_i f_i(x_j, y) \tag{8.4}
\]

\[
\lambda_i = \frac{\log \left( \frac{\text{observed}[i]}{\text{expected}[i]} \right)}{f\#} \tag{8.5}
\]

\[
\lambda_i + = \delta_i \tag{8.6}
\]

Next, GIS computes an update (Eq. 8.5), after which \(\delta_i\) is added to \(\lambda_i\) (eq. 8.6). The algorithm stops when there is no significant change in the \(\lambda_i\) values. This solution is globally optimal as it has been proved that the search space of \(\lambda_i\)s over which we are searching for the final solution is concave leading to the conclusion that every locally optimal solution is globally optimal.

### 8.2.1 Working of FACISME

Equations 8.7, 8.8, and 8.9 are actively used by FACISME in determining the final \(\lambda_i\) values (in each iteration) in conjunction with equations 8.4, 8.5, and 8.6.

\[
\text{expected}[i]+ = \frac{f_i(x_j, y_j) \times e^{s[j,y]}}{z} \tag{8.7}
\]

\[
z = \sum_y e^{s[j,y]} \tag{8.8}
\]

\[
s[j,y]+ = \lambda_i \times f_i(x_j, y_j) \tag{8.9}
\]

The resultant pseudo-code for a single iteration of FACISME is shown in Algorithm 18 and described as follows. First it initializes all \(\lambda_i\)s to 1 and expected values to 0 (Algorithm 18 lines 1–4). Then, iterating over all possible transactions that can be derived by using items present in \(E\) and for each class \(y\), we calculate \(s[j,y]\), taking into account \(\lambda_i\) values from the previous iteration (Algorithm 18 lines 5–14). The presence of each frequent (constraint) itemset \(i \in S\) in a particular transaction \(x_j\), is indicated by \(f_i(x_j, y) = 1\). Based on the \(s[j,y]\) and \(z\) values calculated for each class \(y\), we calculate the \(\text{expected}[i]\) values (Algorithm 18 lines 15–24). If the current iteration is the first iteration, Algorithm 18 line 20 is evoked to calculate \(\text{expected}[i]\). Or else, equation 7 (Algorithm 18 line 18) is used to calculate \(\text{expected}[i]\). Last, for each frequent (constraint) itemset \(i \in S\), we calculate \(\lambda_i\) and \(\delta_i\) (Algorithm 18 lines 25–28). This \(\lambda_i\) is used in the next iteration. We continue iterating over this procedure until \(\text{expected}[i] \approx \text{observed}[i]\) for each \(i\), i.e. till convergence is achieved.
1: if first iteration then
2:   expected[0 . . . F] = 0
3: end if
4: for each training instance j, i.e. all possible transactions using items in E do
5:   if not first iteration then
6:     for each output class y do
7:       s[j, y] = 0
8:       for each i such that \( f_i(x_j, y) \neq 0 \) do
9:         s[j, y] += \( \lambda_i \times f_i(x_j, y) \)
10:     end for
11:   end if
12:   for each output class y do
13:     if not first iteration then
14:       expected[i] += \( f_i(x_j, y) \times e^{s[j, y]} \)
15:     else if first iteration then
16:       expected[i] += \( \frac{1}{|X|} \)
17:     end if
18:   end for
19: end for
20: for each i do
21:   \( \delta_i = \log \left( \frac{\text{observed}[i]}{\sum_{y} e^{s[j, y]} \text{expected}[i]} \right) \)
22: end for
23: \( \lambda_i += \delta_i \)
24: end for

Algorithm 18: Pseudo-code for one iteration of FACISME

8.2.2 Actual Classification using FACISME

Before a given crisp transaction \( x \), containing crisp items, can be classified using a classifier trained by FACISME, we need to run the same pre-processing steps (described in Chapter 3) that we used on the original training dataset before training. We use the fuzzy partitions obtained from pre-processing to transform the crisp attributes present in \( x \) to fuzzy attributes. This transformation process leads to the generation of one or more fuzzy transactions based on the number of fuzzy partitions generated for each crisp numerical attribute. The fuzzy transactions are represented by set \( X' = x'_1, x'_2, \ldots, x'_r \). Our objective is to find the class which best classifies the set \( X' \). For a fuzzy transaction \( x' \in X' \), we first extract all the frequent itemsets \( \in S \) that are subsets of \( x' \). These itemsets are the features of \( x' \). Then, we compute the entropy (equations 8.10 and 8.11) for each class \( y \).

\[
\text{entropy} = \sum_{i} -\log(p_i) \times p_i \tag{8.10}
\]
\[ \text{entropy} = \sum_i e^{\lambda_i} \times \mu \times \text{fuzzy\_support} \]  
\hspace{1cm} (8.11)

Eq. [8.10] is the standard equation for entropy. This equation has been transformed into Eq. [8.11] in order to handle entropy in the fuzzy associative classification context. 

\[-\log(p_i) \text{ in Eq. } 8.10 \text{ is equivalent to } e^{\lambda_i} \text{ in Eq. } 8.11, \text{ (recall that } \lambda_i \text{ is in logarithmic form). And, } p_i \text{ in Eq. } 8.10 \text{ is equivalent to } \mu \times \text{fuzzy\_support}. \]

Because FACISME involves fuzzy logic, the fuzzy membership value \( \mu \) of each fuzzy transaction has to be taken into consideration. Likewise, fuzzy support (as described in Section 4.1) of each frequent itemset (constraint) also needs to be used.

To calculate the entropy, first we find the fuzzy membership \( \mu \) of the current fuzzy transaction using a suitable t-norm (Table 3.1). In this case, we use the \( T_M \) t-norm. Next, for every \( i^{th} \) itemset \( \in S \), which is present in the current fuzzy transaction, we extract its fuzzy support (calculated during fuzzy ARM process) and \( \lambda_i \) value (calculated during training phase of FACISME). Using these three values, we calculate the entropy for each fuzzy transaction \( x' \in X' \), and determine the best class for each \( x' \) based on normalized values of entropy. The overall best class for the set \( X' \) is determined by the normalized values of the number of times each class \( y \) is best in the fuzzy transactions \( \in X' \). We also maintain the total entropy for each class \( y \), so that based on the normalized total entropy, we can select the best class for the whole set \( \in X' \) if more than one class turns out to be best in the same number of fuzzy transactions \( \in X' \).

### 8.2.3 Space and Time Complexities of FACISME

For GIS, the set of all possible transactions \( X \) is stored as a sparse matrix of all non-zero indicator functions for each instance \( j \) and output \( y \). GIS requires \( X \), the \( \lambda \)s of size \( |S| \), as well as the expected and observed arrays, which are also of size \( |S| \). Thus, GIS requires space \( O(|X| + |S|) \), and since \( |X| \) must be at least as large as \( |S| \), this would be \( O(|X|) \). Regarding time complexity, every time Equations 8.7 and 8.8 are used, \( \text{expected}[i] \) is re-calculated. This step takes \( O(|X|) \), and to execute it for each \( i \) would take \( O(|X| \times |S|) \). If the algorithm requires \( m \) iterations for the distribution to converge, the time complexity of GIS can be given as \( O(m \times |X| \times |S|) \).

### 8.2.4 Non-closed Itemsets and FACISME

The maximum entropy model as applied to fuzzy associative classification fails in some cases when a frequent itemset, present in the constraint set \( S \), is “not closed” \([Zak00]\). An itemset is not closed if and only if it has the same frequency as one of its supersets, \( \text{i.e.} \ \text{expected}[s_u] \neq 0, \text{ and } \exists s_v \in S \text{ such that } s_u \subset s_v \land \text{expected}[s_u] = \text{expected}[s_v] \). Thus, a major disadvantage with the maximum entropy model is that there is no solution when the system of constraints has non-closed itemsets in it. Hence, in cases when the system of constraints have non-closed constraints, the exact solution does not exist, and the model parameters will not converge under the GIS algorithm. This is elaborated in \([TP05]\).

Hence, a modified form of the maximum entropy model is used which can accommodate non-closed itemsets. Let \( S' \) be the set of closed constraints in \( S \). The non-closed constraints in \( S \) are only used
to determine whether expected values are 0 or not. Thus, only those constraints $\in S'$ are used for the actual calculation and final classifier building as their expected values are $> 0$. Thus, multiple iterations of GIS are run using $S'$ until convergence is achieved.

8.3 Performance Study

In this section, we describe the performance study model that has been used for testing FACISME using three most-widely-used disparate UCI Machine Learning (ML) datasets, namely iris, breast, and pima. These three datasets are disparate from each other on various facets (like number of attributes, type of attributes – numerical or binary, size of dataset, and type of dataset – dense or sparse). This makes any experimental analysis performed simultaneously on all the three datasets reliable and veracious, and such that its covers a wide range of aspects on which testing of classification accuracy and performance can be done. The performance of FACISME has been compared with that of other state-of-the-art classifiers, and the results of the same are detailed in section 8.3.1. Specifically, the classifiers used for comparisons are:

- Classification based on Predictive Association Rules (CPAR) [YH03]
- Classification based on Multiple Rules (CMAR) [LHP01]
- Classification based on Associations (CBA) [LHM98]
- C4.5 [Qui93]
- Ripper [Coh95]
- Fuzzy Round-Robin Ripper (FR3) [HH09a]
- Fuzzy Unordered Rule Induction Algorithm (FURIA) [HH09b]
- Structural Learning Algorithm in a Vague Environment (SLAVE) [GP99]
- ACME: An Associative Classifier Based on Maximum Entropy Principle [TP05]

For the experiments on FACISME, we downloaded the raw datasets from the UCI ML website. These datasets then underwent the fuzzy pre-processing, mentioned in Chapter 3, before being used to generate fuzzy frequent itemsets. To generate the fuzzy frequent itemsets that are used by FACISME, we have used the fuzzy ARM algorithm described in [MP09] (and in Chapter 4). The minimum support used for the fuzzy ARM process, in order to build the FACISME-based classifier, was 0.2 for all the three datasets. All the other approaches have been implemented by their authors with the parameters that they have used for testing. In all the experiments (on FACISME and other associative classification algorithms), accuracy is measured using 10-fold cross validation. FACISME has been implemented using Java on Windows XP, and the experiments on it were performed on an AMD 2600+ Sempron PC with 512 MB main memory. The results of the remaining algorithms (i.e. except FACISME) are taken from [YH03], [TP05], [HH09a], [HH09b].
8.3.1 Experimental Results

As mentioned above, three standard UCL-ML datasets have been used to illustrate the efficacy of FACISME in terms of accuracy and simplicity of use. Figures 8.1, 8.2, and 8.3 illustrate the experimental results to test accuracy obtained for various classifiers on each of the three UCI-ML datasets. Fig. 8.1 shows the accuracy obtained by each classifier on the iris dataset. FACISME performs the best in terms of accuracy as compared to the other datasets. From Fig. 8.2, we get to know the accuracies for all the classifiers on the breast (breast cancer) dataset. And, FACISME performs nearly as well as the most accurate classifier for this dataset. Finally, Fig. 8.3 depicts the results of the experimental analysis done on pima dataset. Even in this case, FACISME performs second-best as compared to the other classifiers. Thus, the basic inference from this experimental analysis is that FACISME consistently performs very well on the basis of accuracy, and is even the best in two cases.

Each of the classifiers (excepting FACISME) used in this experimental analysis are crisp in nature, and thus use sharp partitioning to convert numerical attributes to binary attributes. Normally 5–6 sharp partitions are used, but these can extend up to 10 or more depending on the nature of the numerical attribute. e.g. the attribute “Age” which is generally in the range “0–100”, can be divided into six sharp partitions, namely “0–15”, “16–30”, “31–45”, “46–60”, “61–75”, “76 and above”. In case of FACISME, fuzzy sets are used instead of sharp partitions, in order to convert numerical attributes to fuzzy partitions. The attribute Age can be divided into three fuzzy partitions, namely “Young”, “Medium-aged”, and “Old”, with each value of age belonging to each of the three fuzzy partitions with some membership value \( \mu \). Thus, the number of sharp partitions that need to be used to handle a numerical attribute is far greater than the number of fuzzy partitions required to do the same. Moreover, this leads to better understanding by the user as there are few well-defined fuzzy partitions with linguistic names/meanings like “Young” and “Old”, as opposed to sharp partitions which are less intuitive and to which a user cannot relate to immediately.

For the current analysis we have used two fuzzy partitions for all numerical attributes throughout all the datasets. As described in section 8.2.3, the time complexity of FACISME is quite high, because of which the time required to train a fuzzy classifier using FACISME is also high. Hence, we have used two fuzzy partitions so as to limit the training time. But, even with just two fuzzy partitions we have achieved quite high accuracies, as described above. More importantly, even with this, we achieved similar (sometimes better results) as other state-of-the-art classifiers used in this experimental analysis. The reason for these high accuracies is that FACISME is based on the theoretically sound and well-established maximum entropy framework.

8.4 Summary of FACISME

In this chapter we have described a new classifier based on the paradigms of association rules mining and fuzzy logic. Though recent classifiers involving association rules have shown their techniques to be accurate, their approaches to build a classifier either manifest unobserved statistical relationships in the dataset, or have no statistical basis at all. FACISME uses the well-known maximum entropy principle and iterative scaling to build a statistical and robust theoretical model for classification. Fuzzy Logic
Figure 8.1 Experimental results on Iris dataset

Figure 8.2 Experimental results on Breast dataset

gives FACISME the capability to deal accurately and efficiently with any type of datasets and domains, and in any kind of environments, which is not necessarily true for traditional crisp classifiers. Thus, using maximum entropy and fuzzy logic, FACISME provides very good accuracy, and can work with all types of data (irrespective of size and type of attributes – numerical or binary).
Figure 8.3 Experimental results on Pima dataset