Fuzzy Clustering, Feature Selection, and Membership Function Optimization

Muriel Bowie

Seminar Paper 2004
DIUF – Department of Informatics
University of Fribourg, Switzerland
muriel.bowie@unifr.ch

Abstract

This paper explores the topic of fuzzy clustering, feature selection, and membership function optimization. Feature selection plays a crucial role for all fuzzy clustering applications, as the selection of appropriate features determines the quality of the resulting clusters. We will show how fuzzy clustering can be applied to data mining problems by introducing some of the most commonly used clustering algorithms. For fuzzy clustering, membership values are assigned to each data point. When the same clustering criteria have to be applied to new data, it can be helpful to have the notion of a membership function which allows to determine the membership values of each data point without repeating the clustering for the new data set. We will therefore present methods to perform membership function optimization based on statistical, genetic and neuro-fuzzy function approximation algorithms in order to provide a short survey of the state-of-the-art.

Keywords: fuzzy clustering, feature selection, membership function approximation, optimization techniques, genetic algorithms.

1. Introduction

Data mining aims at searching through large volumes of data in order to reveal useful information in the form of new relationships, patterns, or clusters, for decision making by a user. Fuzzy sets occur to be a natural tool to perform focused search and help discover dependencies in the data. In data mining, major interests consist in exploring structures and eventual quantification of functional dependencies [23]. This is especially interesting as it prevents from searching for meaningless and trivial patterns in a database. For this purpose, fuzzy clustering algorithms have been developed. Although clustering algorithms are widely used for data partitioning, clustering is a non-trivial task when dealing with large data sets. The success of a clustering process depends on the quality and quantity of the selected features. Therefore, the selection of useful features for clustering has to be tackled very carefully. Obviously, the usefulness of a fuzzy set for modeling a conceptual class largely depends on the appropriateness of its membership function [21]. Therefore, the determination of an accurate and justifiable function for any particular situation is of major concern. Many methods that have been proposed for the estimation and optimization of membership functions have been largely empirical and usually involve the design of experiments on a test population to measure subjective perceptions of membership degrees for some particular class. The accuracy of a membership function is consequently necessarily limited. In this paper, we want to discuss different aspects of fuzzy clustering and their application for data mining purposes. We therefore propose to split the fuzzy clustering process into three subprocesses: feature selection, application of a fuzzy clustering algorithm, and membership function approximation and optimization. First, we will present an overview of different feature selection techniques. Next, we will discuss the topic of fuzzy clustering in general and introduce some fuzzy clustering algorithms. Section 4 is completely dedicated to the concept of membership functions. We will have a closer look at the actual meaning of membership functions and their use for clustering and present different methods for membership function optimization with focus on genetic algorithms and neuro-fuzzy methods. Finally, we will conclude this paper with a discussion and comparison of different aspects and methods presented in the paper.

2. Feature Selection

"An important issue that often confronts data miners in practice is the problem of having too many variables" [16]. It is commonly known that not all variables that are available are likely to be necessary for an accurate discrimination. That means for clustering issues, the consideration of too many variables may lead to a worse model than if some of the variables were removed. Therefore,
in many fuzzy clustering applications, feature selection is the crucial point of success. The goal of feature selection is to select the features in an adequate way in order to encode as much information as possible with respect to the task of interest. Generally, there is a very large set of possible features to compose the feature vectors, and it is subsequently desirable to choose a minimal subset of discriminating features. Although, the approach of using “as many features as available” seems to be intuitively attractive, it is known that this naive approach very often leads to worse results, because the training set size should increase exponentially with the feature vector size [8].

The preprocessing subsequently allows to compute distributions of feature values. Feature selection searches for internal structures in data items. But feature selection is far from being a trivial process, as the selected features determine the clustering quality. The crucial question therefore is [37]: Are the selected features sufficiently representative to enable us to construct realistic clusters or classifiers? Unfortunately, in most data mining problems it is not really obvious which variables are relevant for a given application. Different pattern classification techniques can be used for making the feature values less sensitive to noise and select the most valuable features from a larger set of candidate features. The feature selection can be performed by attributing weights to each available feature. Feature selection is therefore merely a classification problem. For a good overview of classification techniques, see [9].

Evolutionary algorithms are widely used in many optimization problems. They are based on the principles of evolution and are considered as a collective learning process with simple principles: Successful innovations are saved and reused, unsuccessful innovations are ignored. The principle of evolution is based on natural selection and heritage of characteristics. Selection can be described as the continuous process of survival of the fittest in each generation and the vanishing of weak species. Genetic algorithms (GA) are a subgroup of evolutionary algorithms.

The idea of a genetic algorithm is straightforward: First, an initial population of chromosomes are generated, where a chromosome represents a possible solution to a given problem that is represented as a string. Three major operations have been defined for the purpose of use with genetic algorithms: selection, mutation, and crossover. Selection refers to the process of selecting appropriate members of a population which are suited to create an offspring. The selection is often done via a fitness function which determines the fitness of each member of the population. Mostly, members with a very weak fitness are not allowed to create offspring. Crossover is a method to create a child chromosome out of two or more parent chromosomes, which are combined in a predefined way. Mutation is a random process which changes single bits in the chromosomes in order to introduce new genetic information into a population. Sometimes a third
technique called cloning is introduced which produces an exact copy of a chromosome for the next generation. For feature selection with genetic algorithms, a feature subset is represented by a binary string (also called a "chromosome") of length \( m \) (number of available features), with a zero at position \( i \) denoting the absence of feature \( i \), and a one at position \( i \) the presence of the given feature \( i \). A population of chromosomes provides the basis of the algorithm. At each step (generation), the fitness of the chromosomes is evaluated. New chromosomes are produced by the processes of crossover and mutation, where crossover refers to the creation of offspring based on mixing two different parent chromosomes, and mutation is the random perturbation of a single bit in a parent chromosome. The probability of mutation is generally kept fairly low, as there is a risk of destroying useful solutions, nevertheless mutations are necessary in order to inject new genetic information into a population.

![Figure 1: Schematic illustration of a genetic algorithm.](image)

There have been some attempts to solve the problem of feature selection using genetic algorithms. Sliedrecki and Slansky [33] propose an algorithms that corresponds to the description above. A slightly different method has been introduced by Kelly and Davis [20]. Techniques based on genetic algorithms cannot be used to select a fixed number, say \( n \), of features. Some algorithms assign weights each of the different features, which allow to select the \( m \) features with the highest weights. Another technique to maintain a fixed number of one’s in a chromosome, could be the introduction of a new GA operator, called self-crossover. This solution has been published by [25]. The main concept of this operator can be summarized as follows: The self-crossover operator alters the genetic information within a single string selected randomly from the mating pool to produce an offspring. This is done in a way that the stochastic and evolutionary characteristics are preserved. That is, the chromosome is cut into four substrings which are subsequently put together in a different order. This allows to preserve the number of one’s in the string and to change the genome of the chromosome simultaneously. With this method any target chromosome can be generated without introducing mutation.

### 2.1.3. Node Pruning

Node pruning is based on a multilayer feed-forward network combined with a back-propagation learning algorithm. It has been proposed by Mao et al. [22] in 1994. The main idea consists in the definition of a "non-saliency" measure and an algorithm for pruning the least salient nodes in order to reduce the complexity of the network after it has been trained. The pruning of the input nodes is similar to removing the corresponding feature from the feature set. The method has the great advantage to develop simultaneously the optimal feature set and the optimum classifier.

The technique can be described as follows: First, a network is trained with all features, and then the least salient node is removed. The resulting network is subsequently trained again, and the least salient node is selected for deletion. This procedure is repeated until a desired trade-off between classification error and network size is achieved.

Mao et al. [22] define a squared-error cost function which is used for training the network. The saliency of each node is defined as the sum of increase in error. Instead of calculating the saliency directly from the definition, they use a back-propagation method which requires only one pass through the training data.

### 2.2. Comparison of Methods

All methods presented above are based on "hard" feature selection methods, that means, features are either selected or not. Some algorithms which have been developed, assign weights to different features to indicate their significance. In [24] features are weighted for k-means clustering purposes. In general, feature selection algorithms can be divided into two classes: statistical algorithms based on pattern recognition techniques, and those using artificial neural networks. Statistical methods from pattern recognition can be again split into two different categories: optimal techniques which guarantee to find an optimal solution and the so-called suboptimal techniques. which can either yield to a single solution or to many solutions. An overview of feature selection algorithms is given in Figure 2.

In [18] a comparison of different feature selection algorithms is presented. The results show that genetic algorithms are very well suited for medium-sized problems (20-30 dimensions), whereas their performance degrades as the dimensionality increases. This is largely due to the computational effort that is necessary for genetic algorithms. Of course, one could argue that feature selection is generally done in an offline manner where the execution time of an algorithm only plays moderate role. This is true if the feature sets are not too large. However, recent applications require feature selection on data sets with several hundred features, where execution time be-
comes extremely important as it could be impractical to run an algorithm even once on the data set.

Figure 2: An overview of feature selection algorithms. A detailed description of all algorithms can be found in [18] (image reference: [18]).

3. Fuzzy Clustering

The term clustering is generally referred to as a process of grouping objects that are in some sense close to a cluster center. Clustering is therefore a very useful task in the data mining process for identification of groups and interesting distributions and patterns. The major concern in clustering is to discover the organization of patterns into groups in order to reveal similarities and differences and derive conclusions [35]. In clustering no predefined classes exist, as well as no prior knowledge about what kind of relations are desirable among the data is available. That is the reason why the process of clustering is often referred to as an unsupervised process. In contrast to clustering, classification is concerned with assigning a data item to a predefined set of categories.

Classical crisp clustering algorithms generate partitions such that each object is assigned to exactly one cluster. Often, however, objects cannot be assigned to strictly one cluster, because they are located “between” clusters. In these cases, fuzzy clustering methods provide a much more adequate tool for representing data structures. One aspect that is not adequately met by crisp clustering techniques is the handling of uncertainty. This may result in knowledge that is only partially extracted or not extracted at all during the clustering process. In [35] the following facts are presented to illustrate the usefulness of fuzzy clustering in comparison to crisp clustering:

- **Clusters are not overlapping.** That is, each value in a database is classified in at most one cluster. In some cases, the value even falls out of the cluster limits and is not classified at all. However, intentionally we have to admit that it makes sense to classify some values into more than one category.
- **The data values are treated equally in the classification process.** Traditionally, in data mining values are classified in a crisp manner, i.e., the value belongs to a category or not. This techniques don’t consider the fact that sometimes, a value could belong to a category, but only with respect to a certain degree.

It becomes clear from the brief analysis above that a crisp clustering process cannot represent all the states of a real-world system that it attempts to analyze. Results will therefore be either incomplete or wrong. Additionally, they are rarely comprehensible and exploitable. The adaptation of uncertainty measures in this context can be therefore considered as an important issue.

The term fuzziness in the context of clustering refers to the fact that in fuzzy clustering each object is associated with a degree of membership rather than a crisp membership. The most important question to be answered before applying any clustering procedure is [11], which mathematical properties of the data set should be used and how can they identify clusters? Since this question depends on the specific data set, there is no way to describe any universally optimal cluster criteria. Therefore, is is important to note that clustering can result in different partitions of a data set, depending on the criterion considered.

3.1. Clustering Algorithms

"A clustering algorithm is called fuzzy clustering algorithm if and only if it uses a soft competitive (non-crisp)
parameter adaptation strategy”. This is the formal definition of the term fuzzy clustering algorithm presented in [2]. The choice of an appropriate clustering algorithm is crucial for the success of the clustering process. Generally, a clustering algorithm is largely characterized by a proximity measure and a clustering criterion [35]:

- A *proximity measure* is a measure that quantifies how “similar” two feature vectors (i.e., data points) are. Mostly, it is necessary to ensure that all selected features contribute equally to the computation of the proximity measure.

- The *clustering criterion* is generally expressed via a cost function or another type of rules. At this point, one has to take into account the type of clusters which are expected to occur. The clustering criterion has to be therefore selected carefully. A criteria that works well in one case probably performs badly in another and vice versa.

A large amount of clustering techniques have been proposed in the literature. According to the method adopted to define clusters as discussed above, the algorithms can be classified as follows [35]:

1. *Partitional clustering* techniques decompose the data set directly into a set of clusters. These methods try to determine a number of partitions in order to optimize a criterion function.

2. *Hierarchical clustering* methods successively proceed by merging smaller clusters into larger ones, or by splitting large clusters into smaller ones. The result of a hierarchical clustering algorithm consists of a tree of clusters, generally referred to as dendrogram, which outlines the relations between clusters. The dendrogram can subsequently be cut at a desired level in order to obtain a clustering of the data items.

3. *Density-based clustering* attempts to group neighboring objects of a data set into clusters on the basis of a density condition.

4. *Grid-based clustering* is mainly used for spatial data mining. It quantizes the object space into a number of cells in order to perform all operations on the quantized space.

5. *Subspace clustering* techniques improve the quality of clusters by finding subsets of the original space. Applied iteratively, this class of algorithms allows to obtain a coarse-grained clustering of a data space with respect to the clusters of the last iteration.

Although the categories mentioned above are mostly used for crisp clustering algorithms, they can be also adopted for fuzzy clustering issues by implementing an additional layer in the algorithm which is concerned with updating the membership values. Clustering algorithms usually proceed as follows [4]:

- Apply clustering techniques on the output data and then project them onto the input data, generate clusters and select the variables associated with the input-output relations. The clustering method determines clusters on the data space based on Euclidean norm

- Form the membership functions for the variables selected (i.e., determine the shape of the membership functions).

In order to illustrate these principles, we will discuss some of the most widely used fuzzy clustering techniques: the fuzzy C-means algorithm, subtractive clustering, and a genetic algorithm-based clustering method. Other fuzzy clustering algorithms have been proposed in literature, like, for example vector quantization based methods [2], and artificial neural network architectures [9].

3.1.1. The FCM Algorithm

One approach to fuzzy clustering is the fuzzy c-means (FCM) algorithm. It is reasonable to assume that points in the middle region between two cluster centers have a gradual membership of both clusters. The fuzzified c-means algorithm allows each data point to belong to a cluster to a degree specified by a membership grade. This, of course, implies that each data point can belong to several clusters at the same time [19]. The algorithm has been first developed by Dunn [10] and is one of the most commonly used fuzzy clustering algorithms. The idea of Dunn’s algorithm is to minimize a certain objective function. Later Bezdek [3] generalized this fuzzy objective function by introducing a weighting exponent. This final version of the algorithm recognizes spherical clouds of points in a m-dimensional space. The clusters are supposed to all have a similar size. The letter c in the name of the algorithm stands for the number of clusters. Therefore, the number of clusters is fixed and has to be known in advance.

The FCM algorithm partitions a collection of K data points specified by m-dimensional vectors \( u_k \) (where \( k = 1, 2, \ldots, K \)) into c fuzzy clusters. It finds a cluster center in each cluster which minimizes an objective function. The FCM algorithm differs from the hard c-means (see [19] for a description of the hard c-means algorithm) due to the fact that it uses fuzzy partitioning rather than crisp partitioning. In order to implement fuzzy partitioning, the membership matrix \( M \) is allowed to have elements in the range [0,1]. The objective function is de-
scribed as follows:

\[ J(M, c_1, c_2, \ldots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{k=1}^{K} \mu_{ik}^q d_{ik}^2, \]  \(2\)

where \(\mu_{ik}\) is a membership between 0 and 1, \(c_i\) is the center of fuzzy cluster \(i\), \(d_{ik} = \|u_k - c_i\|\) is the Euclidean distance between the \(i\)th cluster center and \(k\)th data point, and \(q \in [0, \infty)\) is a weighted exponent. The membership matrix \(M\) has the following properties:

- the sum of each column is one
- the sum of all elements is \(K\)

The elements of the membership matrix are \(\mu_{ik}\) and they determine the membership value of point \(k\) to cluster \(c_i\). In order for \(J\) to reach a minimum, two conditions have to be met:

\[ c_i = \frac{\sum_{k=1}^{K} \mu_{ik}^q u_k}{\sum_{k=1}^{K} \mu_{ik}^q} \]  \(3\)

and

\[ \mu_{ik} = \frac{1}{\sum_{j=1}^{c} \frac{d_{jk}^2}{d_{ik}^2}^{2/(q-1)}} \]  \(4\)

The algorithm itself consists in iterating through these two defined conditions. It therefore determines the cluster centers \(c_i\) and the membership matrix \(M\) with the following steps ([19]):

1. Initialization of the membership matrix \(M\).
2. Determination of cluster centers \(c_i(i = 1, 2, \ldots, c)\) using (3). The cluster centers can be alternatively initialized before the iteration.
3. Computation of the objective function according to (2). It stops if it is below a specific threshold or if the improvement over the previous iteration is below a certain tolerance.
4. Determine the new \(M\) with (4).
5. Go to step 2.

The fuzzy c-means algorithms is well suited for situations in which the clusters have approximately the same size, shape, and density. However, for cases in which the cluster size and shape may vary, the FCM algorithm is not appropriate. One of the algorithm’s advantage is its simplicity that leads to an efficient computation.

3.1.2. Subtractive clustering

Fuzzy c-means is a supervised algorithm, because it is necessary to define the amount of clusters \(c\) before. If \(c\) is not known in advance, an unsupervised algorithm is necessary for clustering. One such algorithm is called subtractive clustering and it is concerned with finding regions in the feature space with high densities of data points. The data points within a prespecified, fuzzy radius are then subtracted (removed), and the algorithm proceeds its search for a new point with the highest number of neighbors. The iteration continues until all points have been tested. A more detailed description of the subtractive clustering algorithm can be found in [19].
3.1.3. Genetic Algorithms for Fuzzy Clustering

This section discusses the use of genetic algorithms in clustering applications. In contrast to the spherical cluster shapes generated with the fuzzy c-means algorithm, the genetic approach to fuzzy clustering is based on searching for an optimal set of ellipsoids in a given data set. There are two ways to introduce genetic algorithms into fuzzy clustering:

1. Optimizing the performance of a certain clustering model. This approach can be applied to any clustering algorithm, which is likely to be trapped by local extrema. One approach is to initialize the FCM algorithm with the cluster centers found by the genetic algorithm, which could be interesting as FCM is very sensitive to initialization [15].

2. Adapting a genetic algorithm for clustering problems. The idea is to create an appropriate initial population in order to apply a genetic algorithm to a given clustering problem.

In [34] a variable-length genotype based genetic algorithm for clustering is proposed. We will only present a short outline of this algorithm:

In the proposed solution, a single ellipsoid constitutes a completely specified sub-solution to the problem. The idea of this approach is that any set of such sub-solutions can be a possible solution. The fitness of a set of sub-solutions is defined by how well that set solves a given problem. The algorithm works as follows: First, a fixed-length string representation is chosen, analogous to traditional genetic algorithms. Next, an single genotype is allowed to grow and shrink dynamically by the action of domain-independent operators in such a way that the genotype represents a set of completely specified sub-solutions. A major problem for variable-length genetic approaches is the choice of genetic operators that always generate a meaningful offspring. Srikanth et al. defined four such operators: crossover, mutation, insertion, and deletion. A detailed description of these operators as well as the fitness function can be found in [34]. Generally, it can be stated that the proposed approach turns out to be very useful for clustering since it eliminates the restriction of fixing the number of clusters in advance, as it would be necessary for a traditional genetic algorithm.

3.2. Discussion

In this section we have discussed fuzzy clustering as a problem to divide a data set into clusters with appropriate membership values. When the data cannot be represented graphically, it is very difficult for a human observer to determine the correctness of the data partitions. Therefore, the following questions arise [17]:

- If different clustering algorithms are applied to a data set, a variety of partitions is obtained. How can we determine which assignment is correct?
- If the number of clusters is not known, different partitions will lead to a different number of clusters. Which number of clusters will provide an optimal solution?
- Many clustering algorithms search for a certain structure in the data in order to perform the partitioning. But what, if the specific structure is not present in the data set?

The validity of clusters is a very important component of cluster analysis. It is concerned with the question whether the underlying assumptions (like number of clusters and cluster shapes) are appropriate for a given partitioning task. An survey of different cluster validity measures can be found in [17]. The fuzzy c-means algorithm could be extended by replacing the Euclidean distance by another metric in order to define other cluster shapes than only spherical ones. Unfortunately, the fuzzy c-means algorithm cannot be used for an automatic adaptation of the shape for each individual cluster. Gustafson and Kessel [14] have defined an algorithm for this purpose. The main idea is to introduce a matrix that induces for each cluster a norm of its own. A detailed description of the algorithm can be found in [17]. An extension of the Gustafson-Kessel algorithm has been proposed by Gath and Geva [12] which, in contrast to the fuzzy c-means and the Gustafson-Kessel algorithm, is not based on an objective function, but rather on the fuzzification of statistical estimators.

With increasing complexity of the cluster shapes, the clustering algorithms become more and more sensitive to local minima of the objective function, as the number of local minima determines the probability to converge to a local minimum rather than a global one. This especially holds for the Gath-Geva algorithm, whereas the fuzzy c-means is usually not affected by local minima [17]. The cluster algorithms discussed in this section determine cluster centers, cluster parameters, and a membership matrix. The membership matrix contains the membership values of each element of the data set for each of the clusters. Lets now assume that the data set is representative for a given system. Then it would be interesting to determine the membership of each additional data point without running through the same clustering algorithm. In a general case, we should be able to predict the membership of all possible data (e.g. $\mathbb{R}^m$). This can be done by extending the discrete membership matrix to a continuous membership function. Once an approximation of a membership function has been obtained, for example by interpolation, projection, or extension (membership function approximation techniques are discussed
in [17]), it is very often necessary to optimize the resulting function. We will discuss membership function optimization techniques in the next section.

Figure 6: The Gustafson-Kessel algorithm (image reference: [7]).

4. Membership Function Optimization

Since Zadeh introduced the notion of fuzzy sets one of the major difficulties has been concerned with the meaning and measurement of membership functions. By definition, all fuzzy sets are totally characterized by their membership functions. Depending on the interpretation of fuzzyness, the meaning of membership functions changes. Before proceeding into the topic of membership function optimization, we want to first start this section with the formal definition of a membership function, as Zadeh has given it 1965:

A fuzzy (sub)set \( F \) has a membership function \( \mu_F \), defined as a function from a well-defined universe (the referential set), \( X \), into the unit interval \( \mu_F : X \rightarrow [0, 1] \).

The performance of a fuzzy system depends on the rule base and its membership functions. With a given rule base, the membership functions can be optimized in order to improve the overall performance of a fuzzy system. In the past, different methods haven been proposed for the optimization of membership functions. These methods can be divided into two types: derivative-based methods and methods that do not use derivatives. Derivative-based methods include gradient descent, Kalman filtering, the simplex method, least squares, back-propagation, and other numerical techniques. Derivative-free methods are based on genetic algorithms, neural networks, evolutionary programming, geometric methods, fuzzy equivalence relations, and heuristic methods. References to implementations of these techniques can be found in [32].

Derivative-free methods can be applied independently of the actual shape of the objective function. They are therefore more robust for finding a global minimum. Unfortunately, they tend to converge slower than derivative-based methods. Derivative-based methods, on the other hand, tend to converge rather to local minima than to global minima, although the they converge faster. Additionally, they depend on derivatives which makes them more sensitive to the form of the respective objective functions.

4.1. Gradient Descent

Gradient descent approaches to fuzzy membership function optimization are probably the most straightforward implementation of fuzzy function tuning algorithms. Gradient descent is a method for function optimization which is based on the derivate and the idea of steepest descent of a function. As the derivative simply describes the slope of a function, it suffices to move towards the negative direction of the slope in order to reduce the value of the function. The algorithm has the drawback, that the derivative of a function must be available, and it converges to a local minimum rather than a global one. Gradient descent algorithms can be summarized as follows [28]:

1. Compute the derivative of the function. The derivative can be denoted as \( \nabla F(x) \) where \( F \) is the function to be minimized and \( x \) a vector of independent variables.

2. Change the values of \( x \) according to: \( x_{n+1} = x_n - \eta \nabla F(x_n) \), where the subscript \( n \) refers to the iteration number and \( \eta \) to the step size which must be chosen.

3. Iteration until the function converges to a minimum of Function \( F(x) \)

4.2. Genetic Algorithms

As already mentioned, genetic algorithms are well suited for all kinds of optimization problems. It is therefore not surprising that they can also be used for membership function optimization. Genetic algorithms can be used to optimize a fuzzy membership function as follows [26]:

1. First, a population of fuzzy membership functions is created. The fuzzy membership functions can be generated randomly, or they can be created as random perturbations around some nominal functions.
2. Each member of the population has to be represented as a binary string. That means, each function must be translated into an appropriate string representation.

3. Once we have a population of membership functions, the “fitness” of each function has to be evaluated based on some predetermined method.

4. After attributing fitness values to all members of the population, the weakest members are killed. The fittest members then reproduce using two different methods: cloning or crossover. In cloning, the member is reproduced identically, in crossover (mating), two members of the population are selected to form an offspring which is a combination of the parents. Finally, there is a very small probability of mutation in the offspring in order to introduce new genetic information into the new population.

The general procedure is the same as for clustering and feature selection. However, in the case of membership function tuning, the evaluation of the fitness has to be tackled carefully. For example, if we have a set fuzzy clusters and we want to create a membership function for each in such a way as to allow the clustering of any new element, we have to not only consider the discrete membership values of the data points, but we also have to take into account the amount of clusters and the size of the feature space. Membership function optimization might therefore change the initial cluster shape.

4.3. Neuro-fuzzy Function Approximation

An artificial neural network model automatically performs fuzzy clustering when a pattern is allowed to belong to multiple categories with different degrees of membership depending on the neuron’s ability to recognize the input pattern [2]. This would therefore imply that neural networks would be a natural choice for fuzzy clustering. In the last couple of years, several research papers have been published concerning the relation between fuzzy methods and artificial neural networks. The proposed approaches can be categorized into two general classes [6]:

- In cooperative neuro-fuzzy methods fuzzy logic techniques and artificial neural networks work together as independent building blocks. In such approaches, usually neural networks take the results of a neuro-fuzzy operation as input.

- Hybrid methods, in contrast, have such a close coupling between fuzzy-logic and neural networks, that it makes more sense to regard them as an integrative description.

For the purpose of this paper, we will only consider the application of neuro-fuzzy methods to the optimization of membership functions, which is a cooperative neuro-fuzzy method that takes a fixed rule base as input in order to compute an appropriate approximation of the membership function. The rule base can be either available explicitly in parametrized form, or implicitly. Depending on the form of the rule base, the learning process is organized differently. In order to illustrate the tuning of a membership function through a neural network architecture in more depth, we will present a specific algorithm that is described in [6].

4.3.1. A neural network approach to membership function tuning

In this section we will discuss a learning algorithm which can be used for function approximation in Mamdani-Fuzzy systems (see [5] and [6]). Function approximation in the context of neural networks refers to the technique of finding a function that best fits the training data set. The proposed method is based on back-propagation. An error is being determined at the output-side of the fuzzy system and is back-propagated through the architecture. The main information of the error consists in knowing whether a weight of a rule has to be increased or decreased. The size of the error determines the size of the parameter modification. At each iteration, the algorithm calculates the parameter actualization for the membership function of the rule’s consequences and conditions. The stop condition of the algorithm is fulfilled if one of the following conditions hold:

- The total error is less than a certain threshold.
- The total error has reached a local minimum.
- The maximal number of iterations is reached.

It makes sense to observe these conditions in a separate validation set and not in the training set in order to avoid an over-adaptation to the specific learning task.

Suppose, we have a parametrized membership function in triangular form that is part of a rule \( r \) [6]. Then, we have:

\[
\mu_r(x) = \max\{0, 1 - 2\frac{|x - a_r|}{b_r}\}
\]

where \( a_r \) is the maximum point of the membership function in rule \( r \) and \( b_r \) is the width of the triangle. The “consequence parts” of the rules are represented by singletons \( s_r \). If the algebraic production is used for the representation of \( \text{AND’s} \) in the condition part of a rule, then the grades of activation \( \beta^k_{\text{r}} \) are defined by

\[
\beta^k_{\text{r}} = \prod_{i=1}^{N} \mu^k_i(x^k_i),
\]
If \( x \) is the actual input vector. For the output value \( y \) the following holds:

\[
y = \frac{\sum_{r=1}^{R} \beta_r^k s_r}{\sum_{r=1}^{R} \beta_r^k}
\]

(7)

Finally, an incremental gradient descent learning process can be implemented which minimizes the initial error

\[
\epsilon = \frac{1}{2} (d_k - y_k)^2
\]

(8)

For the computation of the parameter modifications a back-propagation algorithm is implemented with the individual learning rates \( \eta_a, \eta_b, \) and \( \eta_y \). The application of the incremental learning rule results in parameter modifications according to the presentation of a tuple \( (x_k, d_k) \). As the triangular function is not differentiable at three points, one could consider not to use training data that can be found at these points. This can only be done if there is a sufficient number of data tuples close to those points as in this case the learning success is hardly disturbed.

### 4.4. Other Optimization Methods

#### 4.4.1. Kalman Filter

The Kalman filter is a set of mathematical equations that provide an efficient computational method to estimate the state of a process. The estimation is based on the minimization of the mean square error. It has been shown that the Kalman filter finds a better solution, and converges faster than gradient descent based methods. The Kalman filter requires that the process to be estimated is linear. Processes that are not linear can be linearized by the extended Kalman filter which is often applied to parameter optimization problems [36]. In [31] a membership function optimization technique using the extended Kalman filter is proposed. In general, the optimization of fuzzy membership function can be viewed as a weighted least-squares minimization problem, where the error vector is the difference between the outputs of the fuzzy system and the target values for those outputs. Before the optimization can be performed, it is necessary to cast the membership function optimization problem into a form suitable for Kalman filtering. The problem is therefore transformed into a nonlinear system architecture where the membership function constitutes the state of the system and the output of the fuzzy system are modeled as the output of a nonlinear system. In this section we will only give a brief outline about how the extended Kalman filter can be applied to fuzzy membership function optimization. A detailed survey of Kalman filtering techniques can be found in [1].

#### 4.4.2. Minimax Filter

Another approach to membership function optimization is based on the Minimax filter, a filter which, in contrast to the Kalman filter, minimizes the worst-case error rather than the mean square error as described above. More precisely, the Minimax filter minimizes the maximum singular value of the transfer function from the noise to the estimation error. While the Kalman filter requires information about the noise properties of the system, the Minimax filter does not require such knowledge. A detailed description of Minimax filtering can be found in [30].

#### 4.4.3. Hybrid Filter

As discussed in the previously, Kalman filtering is an estimation method which minimizes the average estimation error. In contrast, Minimax filtering minimizes the "worst case" estimation error. Unfortunately, both methods have a number of drawbacks [29]:

- In Kalman filtering, it is assumed that the noise properties of a system are known. The Minimax supposes that no information about the system’s noise is available. But sometimes, incomplete information is available on the system’s noise.

- The Kalman filter minimizes the average estimation error, the Minimax filter minimizes the worst-case error. But what, if one prefers to minimize a combination of these?

In [29] a hybrid approach, based on Kalman and Minimax filtering is proposed. Although the concept is very heuristic, it proves to be quite powerful in comparison with the isolated Kalman and the Minimax filters. If \( K^{(2)} \) denotes the steady-state gain of a Kalman filter, and \( K^{(\infty)} \) is the gain of the Minimax filter for the same system. Then the hybrid filter is given by:

\[
K = dK^{(2)} + (1 - d)K^{(\infty)}
\]

with \( d \in [0, 1] \). The parameter \( d \) is the relative weight of the Colman filter performance. If \( d = 0 \) then the hybrid filter is equivalent to the Minimax filter, and if \( d = 1 \), it is equivalent to the Kalman filter. The choice of parameter \( d \) has to be tackled with care in order not to affect the stability of the hybrid filter. The parameter \( d \) actually represents the designer’s confidence in the a priory noise information.

### 4.5. Discussion

The parameter adaptations in the algorithms described above have the drawback that at the end of the learning process the modified fuzzy sets can be different from the actual label of the set. Therefore, the interpretability of the clusters can be affected. However, clustering of large
data sets in a multidimensional feature space is generally not very sensitive regarding the shape of the membership function. For cases which are particularly sensitive, it is not recommended to use membership optimization techniques, but rather to run the same clustering algorithm when new data is inserted, or to use interpolation to extract the membership functions from clustered data.

Genetic optimization is a mechanism that can be applied to any function, independent of its differentiability. It is consequently an attractive approach to membership function optimization. In addition, it converges to a global minimum rather than a local minimum [26].

Given a fuzzy logic system, how can a membership function be determined that will result in best performance? If the membership function is restricted to a specific shape, then it can be easily parametrized by a relatively small number of variables and the membership function optimization problem can be regarded as a parameter optimization problem. This approach for membership function optimization is very often adopted in real-world optimization problems. One drawback of this approach is that it results in membership functions that are generally not sum normal, that means, they don’t add up to one at each data point. In [27] the sum normal constraint is applied to the gradient descent optimization and the Kalman filter optimization of fuzzy membership functions.

We have presented different techniques for membership function optimization of a fuzzy system. We have shown that the optimization problem can be viewed as a system identification problem for a nonlinear dynamic system which is suitable for Kalman filtering. Experiments in [31] suggest that the Kalman filter converges more quickly and finds a better solution than gradient descent-based methods, although it requires more computational effort. We have also presented a hybrid filter which is implemented as a combination of the Kalman and the Minimax filter and is especially useful when a partial knowledge about the system’s noise properties is available.

5. Conclusion

In this paper, we have discussed the process of creating fuzzy sets from a set of data. We have presented methods for feature selection in order to optimize the clustering results, we have shown how data can be clustered using different algorithms and we have discussed the approximation of membership functions. Before proceeding, let us first consider the definition of a fuzzy set which can be found on the Wikipedia ¹ website:

Fuzzy sets are an extension of the classical set theory used in fuzzy logic. A fuzzy set is characterized by a membership-degree function, which maps the members of the universe into the unit interval [0,1]. The value 0 means that the member is not included in the given set, 1 describes a fully included member (this behavior corresponds to the classical sets). The values between 0 and 1 characterize fuzzy members.

Clustering alone does not lead to fuzzy sets as defined above. This is due to the fact that there is no explicit membership function, but only a membership matrix which contains membership values for the specific data points upon which the clustering algorithm has been applied. Once, the membership values for the data set have been obtained, it is possible to produce an approximation of the corresponding membership functions for each cluster by interpolation, projection, or extension. Next, we have presented a number of techniques for membership function optimization in order to improve the expressiveness of the membership function. When we put all these presented methods together, we have all the ingredients which are necessary for fuzzy sets as defined above, what means that we are indeed capable of producing fuzzy sets from an arbitrary data set.

5.1. A Note on Genetic Algorithms

We have introduced the mechanism of genetic algorithms in the paper. Actually, for each of the presented problems, feature selection, fuzzy clustering, and membership function optimization, we have proposed solution based on genetic algorithms. Of course, the technique of genetic algorithms is a very popular and attractive method to solve all kind of problems, but it is important to note that genetic algorithms depend on a multitude of parameters which have to be chosen in advance. This includes among others the selection of an appropriate string representation of the target problem, an initialization method, fitness evaluation, and a termination constraint. The actual solution that is therefore found for a given problem is not based on some scientific or mathematical model, but is rather found by tuning the parameters, which has to be very often done manually. Artificial neural networks, in contrast, do not depend on so many parameters which have to be chosen in advance as their strength lies in the fact that they can automatically adapt the parameters, i.e. their weights and thresholds, by learning mechanisms.

In this paper, we have presented a set of methods for feature selection, fuzzy clustering, and membership function optimization. We have shown which techniques can be used under which circumstances and have outlined the advantages and drawbacks of each method. Of course, there are other techniques that have been adopted for the same tasks, and it was not the objective of this paper to give an inventory of all available methods, but rather to present a survey of the state-of-the art in the field.

¹http://en.wikipedia.org
6. References


