Fuzzy Sets and Pattern Recognition

1. Introduction

Previously we have discussed using various distance-based techniques for clustering and classifying features obtained from sensor outputs. These techniques are based on looking at the mean and covariance of the features to determine if they belong in a given class. In this section of the course, we will examine the notions of crisp and fuzzy classes. We will also examine some relatively simple techniques for determining the non-linear transfer functions that map features into classes. These transfer functions enable the HCI designer to develop systems that can find patterns in the data obtained from input sensors and map these patterns to specific actions which can control the operation of the computer.

For example, as was discussed in the first homework assignment, we might want to recognize whether someone is nodding their head to mean "yes" or shaking their head to mean "no" when interacting with a computer. Two accelerometers can be used to obtain the data on head movement. However, it is up to the computer to find patterns in the accelerometer data which correspond to the two head gestures. This is a relatively straight forward task as there would only need to be two features (the output level of each accelerometer) and two classes (the two head gestures). The transfer function between features and class would probably be quite obvious. If the forward facing accelerometer's output is high and the side facing accelerometer's output is low, then the individual is nodding. The opposite sensor outputs would indicate a shake..

What if, on the other hand, one wanted to use sign language to communicate with the computer. Many sensor outputs which measured finger, hand, and arm positions would be needed. All or most of these outputs would be used to calculate important features of the gesture. The computer would then have to use some or all of these features to determine which gesture was made. This is an extremely complicated problem with the possibility that an individual might not use exactly the same gesture for a given sign every time. It is also quite likely that each individual might make slightly different gestures for the same sign. The transfer function between sensor features and a sign may be quite different and might have to be changed for each individual.

Pattern recognition using fuzzy sets, which is discussed in this section, is a technique for determining such transfer functions. A good overview of the material discussed is "A Review of Probabilistic, Fuzzy, and Neural Models for Pattern Recognition" by James C. Bezdek in the *Journal of Intelligent and Fuzzy Systems*, vol.1 (1), pgs. 1-25, 1993.

2. What are fuzzy sets and who uses them?

Before talking about how to use fuzzy sets for pattern classification, we must first define what we mean by fuzzy sets. A great source of information on fuzzy sets and fuzzy logic can be found in a collection of frequently asked questions and corresponding answers. http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html

2.1 Fuzzy sets vs. crisp sets

Crisp sets are the sets that we have used most of our life. In a crisp set, an element is either a member of the set or not. For example, a jelly bean belongs in the class of food known as candy. Mashed potatoes do not.

Fuzzy sets, on the other hand, allow elements to be *partially* in a set. Each element is given a degree of membership in a set. This membership value can range from 0 (not an element of the set) to 1 (a member of the set). It is clear that if one only allowed the extreme membership values of 0 and 1, that this would actually be equivalant to crisp sets. A membership function is the relationship between the values of an element and its degree of membership in a set. An example of membership functions are shown in Figure 2-1. In this example, the sets (or classes) are numbers that are negative large, negative medium, negative small, near zero, positive small, positive medium, and positive large. The value, μ, is the amount of membership in the set.

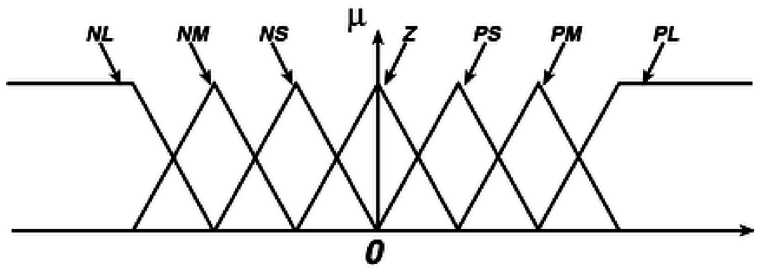

Figure 2-1: Membership Functions for the Set of All Numbers (N = Negative, P = Positive, L = Large, M = Medium, S = Small)

2.2 Applications

Fuzzy sets are appropriate for pattern classification because a given gesture or pattern may in fact have partial membership in many different classes. Several companies already have products based on fuzzy pattern recognition:

1. Hand Writing Recognition: CSK, Hitachi

2. Hand Printed Character Recognition: Sony
3. Voice Recognition: Ricoh, Hitachi

3. Relationship of fuzzy classifiers to statistical classifiers and neural classifiers

3.1 Fuzzy vs. Statistical

One of the most common questions regarding fuzzy set classification is how does it relate to statistical classification. What is the difference between the degree of membership (also known as possibility) in the set and the probability of being in that set? Once again refer to the fuzzy FAQ.

A good example that demonstrates the conceptual difference between statistical and fuzzy classification is the one given by Bezdek in the reference mentioned previously. In the example, a person who is dying of thirst in the desert is given two bottles of fluid. One bottle's label says that it has a 0.9 membership in the class of fluids known as non-poisonous drinking water. The other bottle's label states that it has a 90% probability of being pure drinking water and a 10% probability of being poison. Which bottle would you choose?

In the example, the "probability bottle" contains poison. This is quite plausible since there was a 1 in 10 chance of it being poisonous. The "fuzzy bottle" contains swamp water. This also makes sense since swamp water would have a 0.9 membership in the class of non-poisonous fluids. The point is that probability involves crisp set theory and does not allow for an element to be a partial member in a class. Probability is an indicator of the frequency or likelihood that an element is in a class. Fuzzy set theory deals with the similarity of an element to a class.

Both are valid approaches to the classification problem. If we were to classify someone as "old", fuzzy membership makes much more sense than probability. If we were to classify the outcome of a coin flip, probability makes much more sense.

3.2 Fuzzy vs. Neural

Both fuzzy systems and neural networks attempt to determine the transfer function between a feature space and a given class. (Note: if the reader is unfamiliar with neural networks, an excellent overview by Dr. Leslie Smith can be found here. http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html) Both can be automatically adapted by the computer in an attempt to optimize their classification performance.

One difference between the two methods is that the membership functions of a fuzzy classifier can be initialized in a state close to the correct solution. What this means is that a fuzzy classifier can be set up by a skilled HCI designer to do a pretty good job of classification even before the classifier is adjusted by the computer. A neural network, however, can only be initialized in a random state. Thus, the training of the computer to optimize the classifier is usually much faster with a fuzzy classifier than a neural network classifier.

The problem with a fuzzy system is it is difficult to deal with too many features, membership functions, and/or rules. Neural networks, are highly suited for large amounts of features and classes.

4. Fuzzy inference systems

A fuzzy inference system (FIS) is a system that uses fuzzy set theory to map inputs (*features* in the case of fuzzy classification) to outputs (*classes* in the case of fuzzy classification). Two FIS's will be discussed here, the Mamdani and the Sugeno.

4.1 Fuzzy inference systems (Mamdani)

An example of a Mamdani inference system is shown in Figure 4-1. To compute the output of this FIS given the inputs, one must go through six steps:

1. determining a set of fuzzy rules
2. fuzzifying the inputs using the input membership functions,
3. combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,
4. finding the consequence of the rule by combining the rule strength and the output membership function,
5. combining the consequences to get an output distribution, and
6. defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

The following is a more detailed description of this process.

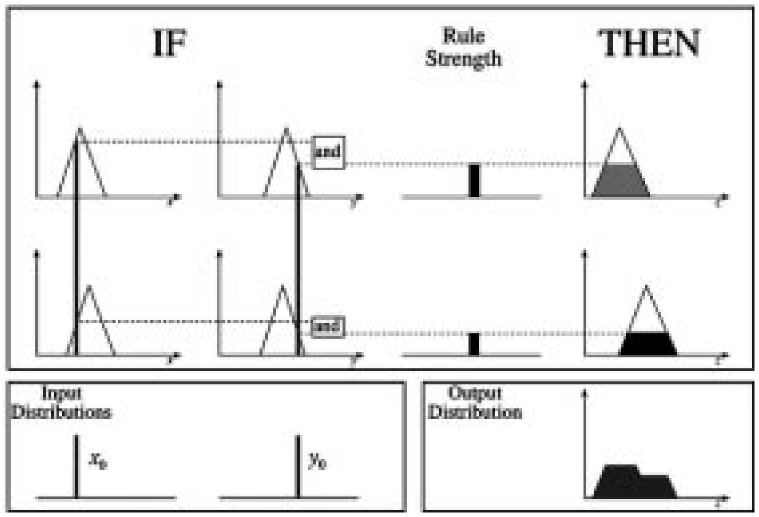

Figure 4-1: A two input, two rule Mamdani FIS with crisp inputs

4.1.1 Creating fuzzy rules

Fuzzy rules are a collection of linguistic statements that describe how the FIS should make a decision regarding classifying an input or controlling an output. Fuzzy rules are always written in the following form:

if *(input1 is membership function1)* ***and/or*** *(input2 is membership function2)* ***and/or*** *….* ***then*** *(output$_n$ is output membership function$_n$).*

For example, one could make up a rule that says:

if *temperature is high* ***and*** *humidity is high* ***then*** *room is hot.*

There would have to be membership functions that define what we mean by high temperature (input1), high humidity (input2) and a hot room (output1). This process of taking an input such as temperature and processing it through a membership function to determine what we mean by "high" temperature is called fuzzification and is discussed in section 4.1.2. Also, we must define what we mean by "and" / "or" in the fuzzy rule. This is called fuzzy combination and is discussed in section 4.1.3.

4.1.2 Fuzzification

The purpose of fuzzification is to map the inputs from a set of sensors (or features of those sensors such as amplitude or spectrum) to values from 0 to 1 using a set of input membership functions. In the example shown in Figure 4-1, their are two inputs, x_0 and y_0 shown at the lower left corner. These inputs are mapped into fuzzy numbers by drawing a line up from the inputs to the input membership functions above and marking the intersection point.

These input membership functions, as discussed previously, can represent fuzzy concepts such as "large" or "small", "old" or "young", "hot" or "cold", etc. For example, x_0 could be the EMG energy coming from the front of the forearm and y_0 could be the EMG energy coming from the back of the forearm. The membership functions could then represent "large" amounts of tension coming from a muscle or "small" amounts of tension. When choosing the input membership functions, the definition of what we mean by "large" and "small" may be different for each input.

4.1.3 Fuzzy combinations (T-norms)

In making a fuzzy rule, we use the concept of "and", "or", and sometimes "not". The sections below describe the most common definitions of these "fuzzy combination" operators. Fuzzy combinations are also referred to as "T-norms".

4.1.3.1 Fuzzy "and"

The fuzzy "and" is written as:

$$u_{A \cap B} = T(u_A(x), u_B(x))$$

where μ_A is read as "the membership in class A" and μ_B is read as "the membership in class B". There are many ways to compute "and". The two most common are:

1. Zadeh - min(u_A(x), u_B(x)) This technique, named after the inventor of fuzzy set theory simply computes the "and" by taking the minimum of the two (or more) membership values. This is the most common definition of the fuzzy "and".
2. Product - u_a(x) times u_b(x)) This techniques computes the fuzzy "and" by multiplying the two membership values.

Both techniques have the following two properties:

T(0,0) = T(a,0) = T(0,a) = 0

T(a,1) = T(1,a) = a

One of the nice things about both definitions is that they also can be used to compute the Boolean "and". Table 1 shows the Boolean "and" operation. Notice that both fuzzy "and" definitions also work for these numbers. The fuzzy "and" is an extension of the Boolean "and" to numbers that are not just 0 or 1, but between 0 and 1.

Input1 (A)	Input2 (B)	Output (A "and" B)
0	0	0
1	0	0

0	1	0
1	1	1

Table 1: The Boolean "and"

4.1.3.2 Fuzzy “or”

The fuzzy “or” is written as:

$u_{A \cup B} = T(u_A(x), u_B(x))$

Similar to the fuzzy “and”, there are two techniques for computing the fuzzy “or”:

1. Zadeh - max($u_A(x)$, $u_B(x)$) This technique computes the fuzzy “or” by taking the maximum of the two (or more) membership values. This is the most common method of computing the fuzzy “or”.
2. Product - $u_A(x) + u_B(x) - u_A(x)\, u_B(x)$ This technique uses the difference between the sum of the two (or more) membership values and the product of the membership values.

Both techniques have the following properties:

T(a,0) = T(0,a) = a

T(a,1) = T(1,a) = 1

Similar to the fuzzy “and”, both definitions of the fuzzy “or” also can be used to compute the Boolean “or”. Table 2 shows the Boolean “or” operation. Notice that both fuzzy “or” definitions also work for these numbers. The fuzzy “or” is an extension of the Boolean “or” to numbers that are not just 0 or 1, but between 0 and 1.

Input1 (A)	Input2 (B)	Output (A “or” B)
0	0	0
1	0	1
0	1	1
1	1	1

Table 2: The Boolean "or"

4.1.4 Consequence

The consequence of a fuzzy rule is computed using two steps:

1. Computing the rule strength by combining the fuzzified inputs using the fuzzy combination process discussed in section 4.1.3. This is shown in Figure 4-1. Notice in this example, the fuzzy “and” is used to combine the membership functions to compute the rule strength.
2. Clipping the output membership function at the rule strength. Once again, refer to Figure 4-1 to see how this is done for a two input, two rule Mamdani FIS.

4.1.5 Combining Outputs into an Output Distribution

The outputs of all of the fuzzy rules must now be combined to obtain one fuzzy output distribution. This is usually, but not always, done by using the fuzzy “or”. Figure 4-1 shows an example of this. The output membership functions on the right hand side of the figure are combined using the fuzzy “or” to obtain the output distribution shown on the lower right corner of the figure.

4.1.6 Defuzzification of Output Distribution

In many instances, it is desired to come up with a single crisp output from a FIS. For example, if one was trying to classify a letter drawn by hand on a drawing tablet, ultimately the FIS would have to come up with a crisp number to tell the computer which letter was drawn. This crisp number is obtained in a process known as defuzzification. There are two common techniques for defuzzifying:

1. Center of mass - This technique takes the output distribution found in section 4.1.5 and finds its center of mass to come up with one crisp number. This is computed as follows:

$$z = \frac{\sum_{j=1}^{q} Z_j u_c(Z_j)}{\sum_{j=1}^{q} u_c(Z_j)}$$

where z is the center of mass and u_c is the membership in class c at value z_j. An example outcome of this computation is shown in Figure 4-2.

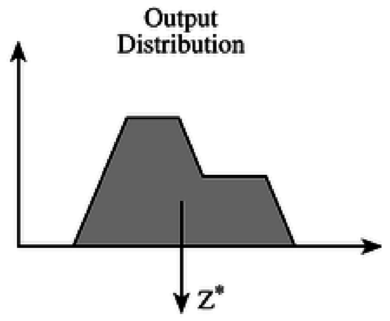

Figure 4-2: Defuzzification Using the Center of Mass

2. Mean of maximum - This technique takes the output distribution found in section 4.1.5 and finds its mean of maxima to come up with one crisp number. This is computed as follows:

$$z = \sum_{j=1}^{l} \frac{z_j}{l}$$

where z is the mean of maximum, z_j is the point at which the membership function is maximum, and l is the number of times the output distribution reaches the maximum level. An example outcome of this computation is shown in Figure 4-3.

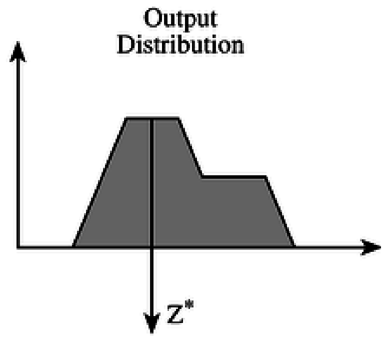

Figure 4-3: Defuzzification Using the Mean of Maximum

4.1.7 Fuzzy Inputs

In summary, Figure 4-1 shows a two input Mamdani FIS with two rules. It fuzzifies the two inputs by finding the intersection of the crisp input value with the input membership function. It uses the minimum operator to compute the fuzzy "and" for combining the two fuzzified inputs to obtain a rule strength. It clips the output membership function at the rule strength. Finally, it uses the maximum operator to compute the fuzzy "or" for combining the outputs of the two rules.

Figure 4-4 shows a modification of the Mamdani FIS where the input y_0 is fuzzy, not crisp. This can be used to model inaccuracies in the measurement. For example, we may be measuring the output of a pressure sensor. Even with the exact same pressure applied, the sensor is measured to have slightly different voltages. The fuzzy input membership function models this uncertainty. The input fuzzy function is combined with the rule input membership function by using the fuzzy "and" as shown in Figure 4-4.

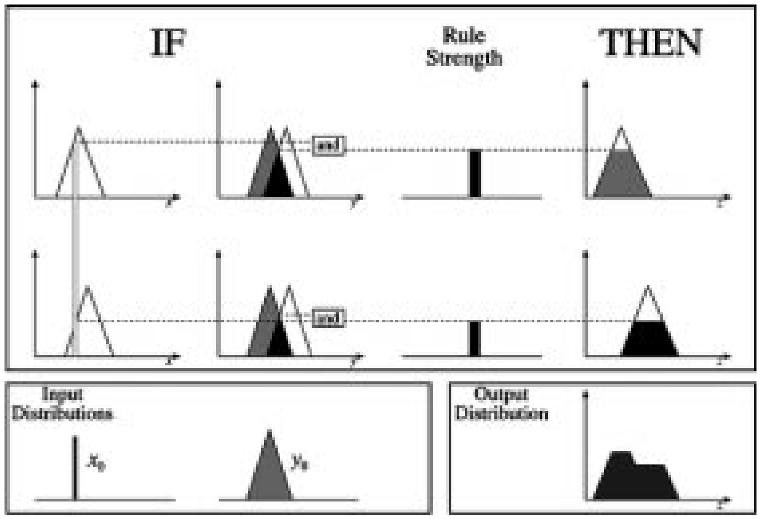

Figure 4-4: A two Input, two rule Mamdani FIS with a fuzzy input

4.2 Fuzzy Inference System (Sugeno)

The Sugeno FIS is quite similar to the Mamdani FIS discussed in section 4. The primary difference is that the output consequence is not computed by clipping an output membership function at the rule strength. In fact, in the Sugeno FIS there is no output membership function at all. Instead the output is a crisp number computed by multiplying each input by a constant and then adding up the results. This is shown in Figure 4-5. "Rule strength" in this example is referred to as "degree of applicability" and the output is referred to as the "action". Also notice that there is no output distribution, only a "resulting action" which is the mathematical combination of the rule strengths (degree of applicability) and the outputs (actions).

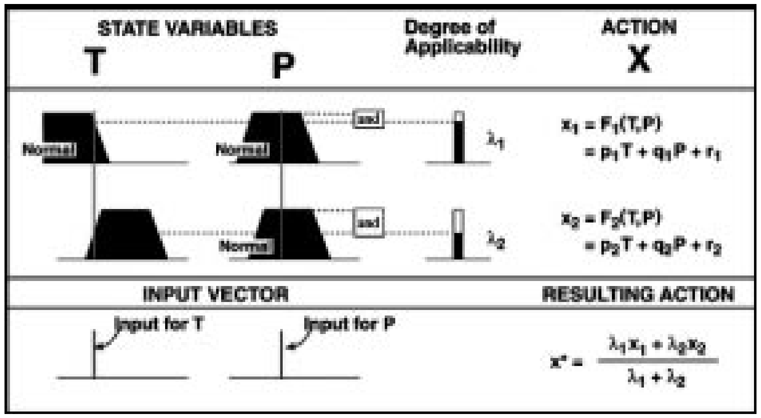

Figure 4-5: A two input, two rule Sugeno FIS (p_n, q_n, and r_n are user-defined constants)

One of the large problems with the Sugeno FIS is that there is no good intuitive method for determining the coefficients, p, q, and r. Also, the Sugeno has only crisp outputs which may not be what is desired in a given HCI application. Why then would you use a Sugeno FIS rather than a Mamdani FIS? The reason is that there are algorithms which can be used to automatically optimize the Sugeno FIS. One of these algorithms is discussed in section 5.1.2

In classification, p and q can be chosen to be 0 and r can be chosen to be a number that corresponds to a particular class. For example, if we wanted to use the EMG from a person's forearm to classify which way his/her wrist was bending, we could assign the class "bend_inward" to have the value r = 1. We could assign the class "bend_outward" to have the value r=0. Finally, we could assign the class "no_bend" to have the value r=0.5.

5. Adaptive Fuzzy

5.1.1 Artificial (Computational) Neural Network

The HCI designer can use neural networks to implement many aspects of the FIS. (Note: if the reader is unfamiliar with neural networks, an excellent overview by Dr. Leslie Smith can be found here. http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html) For example, a neural network can be used to implement:

1. Membership function
2. Fuzzy rules
3. Defuzification

5.1.2 Adaptive neuro-fuzzy inference system (ANFIS)

As mentioned in section 4.2, a the parameters of a Sugeno FIS can be optimized automatically using a computer algorithm. One such algorithm, the adaptive neuro-fuzzy inference system (ANFIS), adapts the parameters of the FIS using neural networks. The original paper on ANFIS can be found in "ANFIS: Adaptive Network Based Fuzzy Inference Systems," by J - S. R. Jang in *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, May 1993.

To use ANFIS, the HCI designer needs to perform the following steps:

1. Design a Sugeno FIS approriate for the classification problem.
2. Hand optimize the FIS given actual input classification data.
3. Set up training and testing matrices. The training and testing matrices will be composed of inputs and the desired classification corresponding to those inputs.
4. Run the ANFIS algorithm on the training data.
5. Test the results using the testing data.