• One field is concerned with systems that recognize patterns with a classification scheme that has already been determined.
  
  – Example: Character recognition, the categories (26 letter of alphabet) are already known.
  
  – Referred to as pattern classification.
Pattern Recognition Systems

- Other field is concerned with obtaining a classifier for a set of patterns where the categories are not initially known.
  - Example: Clustering data concerning patients to see whether patients can be grouped in a way that helps indicate diagnoses.
  - Referred to as Pattern clustering

Fuzzy Systems ToolBox, Mark Beale and Howard Demuth
Pattern Classification

- Attempt to pick the correct category when presented with a pattern.
  - Solution is very problem specific
- Character recognition will be different from recognizing the tank in the battle field
Design a Fuzzy Pattern Classifier

• Pattern Features
  – Create fuzzy sets that represent features in the data that are considered important.
  • Example: ratio of the character’s height to its width may be important while the size of a character is not!
Design a Fuzzy Pattern Classifier

• Defining the System
  – fuzzy sets are used to convert crisp inputs to grades that are then combined by a pattern classifier to obtain grades for each of the possible classification categories.
  • Highly problem dependent!
Design a Fuzzy Pattern Classifier

• Testing the System
  – use known and unknown test sets.
  – Typically, the system will not perform as well as desired at first.

• One may have to test a fuzzy classifier
  – find why it gave a wrong result,
  – make changes to improve the classifier.

Fuzzy Systems ToolBox, Mark Beale and Howard Demuth
Application: Character Recognition

- Goal: design a system that will pick the correct character when it is given a bit map representation of a printed or written input character.
  - Example: classifying the ten digits 0 through 9.
Pattern Features

- INPUT
  - bit map
    - 25 horizontal X 37 vertical pixels (925 pixel field)
    - each pixel have a 0 or 1 associated with it.
Pattern Features

- Creating a rule set dealing with 925 inputs would be extremely difficult and time consuming.
- Fuzzify the input, resulting in a compact summary of the overall state of the bit map.
Fuzzy sets for the Horizontal Position

- Define five triangular-shaped fuzzy sets for the horizontal position.
  - $x_S = 0:24$;
  - $x_L = \text{eqshape}(3,x_S,5)$;
  - $x_G = \text{shapeval}(x_L,x_S)$;
Fuzzy sets for the Vertical Position

- Define seven triangular-shaped fuzzy sets for the vertical position.
  - \( y_S = 0:36; \)
  - \( y_L = eqshape(3, y_S, 7); \)
  - \( y_G = shapeval(y_L, y_S); \)
Fuzzify the input pixel

- Given any pixel \((\text{horizontal } x \text{ and vertical } y)\), the membership of that pixel can be calculated for each of 5 horizontal and 7 vertical regions.
  - \(G1 = \text{fuzzyval} (xS,xG,x)\);
  - \(G2 = \text{fuzzyval} (yS,yG,x)\);
Crate 5 x 7 regions of the 2-D bitmap

- Find all combinations of intersections between the two groups of fuzzy sets
  \[ G_3 = \text{allcomb('min', G1, g2)}; \]
  - \( G_3 \) will be a 35-element column vector of grades associated with the 35 regions of the 2-D bit map
Multiple Pixels

• If vectors X and Y represent the input pixels, then
  – \( G1 = \text{fuzzyval}(xS, xG, X) \);
  – \( G2 = \text{fuzzyval}(yS, yG, Y) \); and
  – \( G3 = \text{allcomb}(\text{‘min’}, G1, G2) \);

• \( G3 \) represents a 35-element column for every pixel crossed by all of the input pixels
Single Grade for each 5 x 7 regions

- Taking the union of all grades associated with each square region:
  \[ G = \text{or} (G_3')' \]

- We now have 35 grades, each grade measuring to what extent the user’s drawn character passed through a different region.

- 925-pixel bit map ----> 35 grades!
Defining the System

• Once the good features have been found, one can define a system to transform the feature grades into the grades for each class.
Defining the System

- **Zero:**
  - $G_0 = \text{and}([\text{or}(G[[2, 7]]); \text{or}(G[[3, 8]]); \text{or}(G[[4, 9]]));$
  - $\text{or}(G[[11, 12]]); \text{or}(G[[16, 17]]); \text{or}(G[[21, 22]]);$
  - $\text{or}(G[[14, 15]]); \text{or}(G[[19, 20]]); \text{or}(G[[24, 25]]);$
  - $\text{or}(G[[27, 32]]); \text{or}(G[[28, 33]]); \text{or}(G[[29, 34]]);$
  - $\text{not}(G[[13, 18, 23]]));$
Defining the System

- One:
Defining the System

• **One:**
  • G1 = or( and( G([5 9 18 27 31]);
                  not(G([1 2 3 6 7 11 25 29 30 33 34 35]));
                and( G([1 7 18 29 35]);
                  not(G([3 4 5 9 10 15 21 26 27 31 32 33]));
                and( G([1 6 11 16 21 26 31]);
                  not(G([3 4 5 8 10 13 15 18 20 23 25 28 30 33 34 35]));
                and( G([2 3 4 5]);
                  not(G([6 7 10 11 12 15 16 17 20 25 30 35]));
                  or( G([8 9]));
                or( G([13 14]));
                or( G([18 19]));
                or( G([23 24]));
                or( G([28 29]));
                or( G([33 34]));
          ));
Defining the System

- **Two:**
  - \( G_2 = \text{and}(\text{somewhat}(G(2)); \ G([3 \ 32 \ 33 \ 34])); \)
  - \( \text{or}(G([6 \ 7])); \ \text{or}(G([30 \ 25])); \ \text{or}(G([4 \ 5])); \)
  - \( \text{not}(\text{or}([G(22 \ 28]); \ \text{and}(G([10 \ 15]));])]); \)
Defining the System

- **Three:**
- \( G3 = \text{and}(\text{G}([2\ 3\ 19\ 32\ 33\ 34])); \)
- \( \text{or}(\text{G}([9\ 10])); \)
- \( \text{or}(\text{G}([14\ 15])); \)
- \( \text{not}(\text{G}([5\ 16\ 21])); \)
Defining the System

- Four:
  - \[ G_4 = \text{and}(\text{or}(\text{and}(G([4, 14, 19, 29])); \text{and}(G([5, 15, 20, 30]))) \]
  - \[ \text{and}(G([3, 13, 18, 28])); \]
  - \[ \text{or}((\text{and}(G([12, 14, 15])); \text{and}(G([17, 19, 20])); \]
  - \[ \text{and}(G([22, 24])); \]
  - \[ \text{not}((\text{G([1, 2, 6, 7])); \text{and}(G([32, 33, 34])); \})}); \]
Once the grades for each class have been found, they can be combined into a single grade vector, \( c_G \)

\[
c_G = [G_0 \ G_1 \ G_2 \ G_3 \ G_4 \ G_5 \ G_6 \ G_7 \ G_8 \ G_9];
\]

Then the maximum grade characters can be found

\[
c = c_S(\text{find}(c_G \geq \text{max}(c_G)));
\]
Testing the System

- Demo
  - Matlab: numrec