7. Examples

Introduction

This notebook contains a number of simple problems which can be dealt with using fuzzy sets. We provide a solution for each problem, but encourage you to try to come up with your own solutions to some of the problems.

If the Fuzzy Logic Pack isn't already loaded, it should be loaded before beginning the problems in this notebook. We can load all of the necessary routines with the following command.

```
Needs["FuzzyLogic`Master`"]
```

If the package loading command fails, verify that the directory containing the Fuzzy Logic directory is on Mathematica's $Path. You can use the command AppendTo[$Path, "directory"]. For convenience, this should be in your init.m file.

Example 1. Classifying Houses

Problem. A realtor wants to classify the houses he offers to his clients. One indicator of comfort of these houses is the number of bedrooms in it.

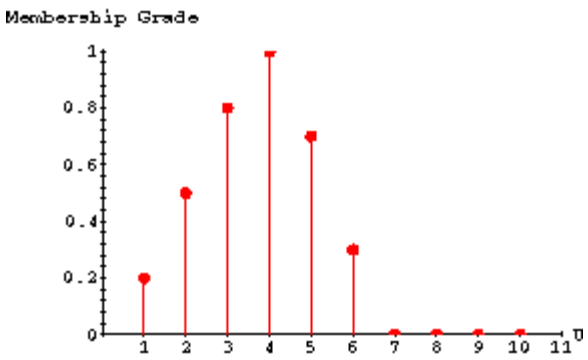
Let the available types of houses be represente by the following set.

U = {1,2,3,4,5,6,7,8,9,10}

The houses in this set are described by u number of bedrooms in a house. The realtor wants to describe a "comfortable house for a 4-person family," using a fuzzy set.

Solution. The fuzzy set "comfortable type of house for a 4-person family " may be described using a fuzzy set in the following manner.

```
HouseForFour = FuzzySet[{{1,.2},{2,.5},{3,.8},{4,1},
  {5,.7},{6,.3}},UniversalSpace->{1,10}];
FuzzyPlot[HouseForFour,ShowDots->True];
```



Example 2. Representing Age

Problem 2-1. Fuzzy sets can be used to represent fuzzy concepts. Let U be a reasonable age interval of human beings.

U = {0,1,2,3,...100}.

Solution 2-1. This interval can be interpreted with fuzzy sets by setting the universal space for age to range from 0 to 100.

```
SetOptions[FuzzySet,UniversalSpace->{0,100}];
```

Problem 2-2. Assume that the concept of "young" is represented by a fuzzy set Young whose membership function is given by the following fuzzy set.

```
Young = FuzzyTrapezoid[0,0,25,40];
```

The concept of "old' can also be represented by a fuzzy set, Old, whose membership function could be defined in the following way.

```
Old = FuzzyTrapezoid[50,65,100,100];
```

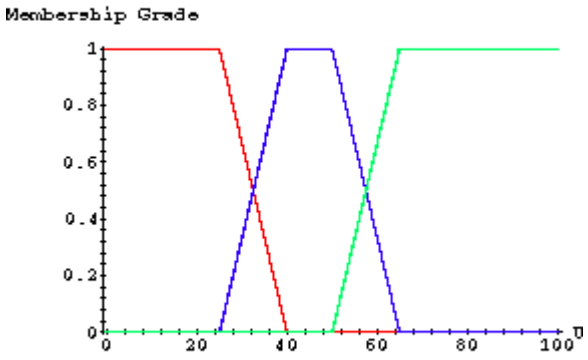
We would now like to define the concept of middle-aged to be neither young nor old. We would like to do this by using fuzzy operators from the Fuzzy Logic Pack.

Solution 2-2. We can find a fuzzy set to represent the concept of middle-aged by taking the intersection of the complements of our Young and Old fuzzy sets.

```
MiddleAged = Intersection[Complement[Young],
  Complement[Old]];
```

We can now see a graphical interpretation of our age descriptors by using the FuzzyPlot command.

```
FuzzyPlot[Young,MiddleAged,Old,PlotJoined->True];
```

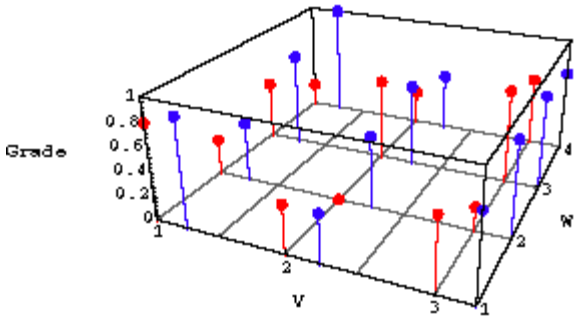


From the graph, you can see that the intersection of "not young" and "not old" gives a reasonable definition for the concept of "middle-aged".

Example 3. Finding the Disjunctive Sum

Problem. Find the disjunctive sum of the two fuzzy relations defined here.

```
RMat = {{.8,.3,.5,.2},
        {.4, 0,.7,.3},
        {.6,.2,.8,.6}};
Smat = {{.9,.5,.8, 1},
        {.4,.6,.7,.5},
        {.7,.8,.8,.7}};
R = FromMembershipMatrix[RMat];
S = FromMembershipMatrix[Smat];
FuzzyPlot3D[R,S,ShowDots->True];
```



The disjunctive sum of fuzzy relations R and S in VxW, can be found with the following formula.

DisSum = Union[Intersection[R,S'],Intersection[R',S]]

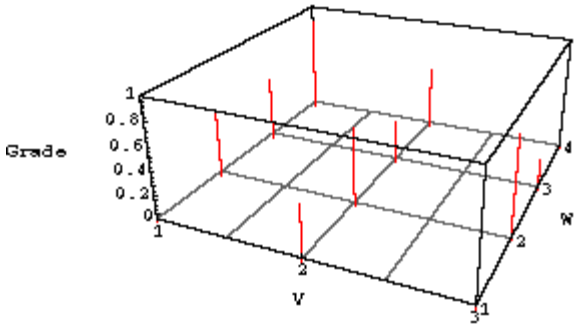
The disjunctive sum is thus a relation in VxW that has the following property.

For all (v,w) in VxW, DisSum(v,w) = Max[Min[R(v,w),1-S(v,w)], Min[1-R(v,w),S(v,w)]]

For fuzzy relations R and S, the disjunctive sum can be computed as follows.

Solution. We can find disjunctive sum of the two fuzzy relations by using the formula serived earlier and some of the functions of the Fuzzy Logic Pack.

```
DisSum = Union[Intersection[R,Complement[S]],
  Intersection[Complement[R],S]];
FuzzyPlot3D[DisSum];
```




```
ToMembershipMatrix[DisSum]
```

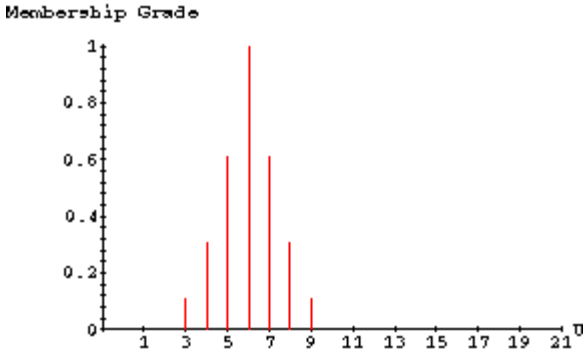
0.2	0.5	0.5	0.8
0.4	0.6	0.3	0.5
0.4	0.8	0.2	0.4

Example 4. Natural Numbers

Problem. Suppose you are asked to define the set of natural numbers close to 6. There are a number of different ways in which you could accomplish this using fuzzy sets.

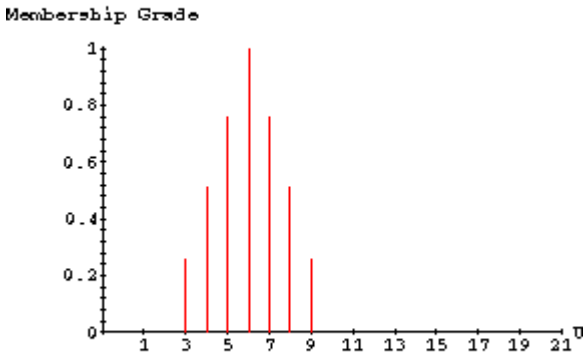
Solution 1. One solution would be to manually create a fuzzy set which describes the numbers near 6. This can be done like the following:

```
SetOptions[FuzzySet,UniversalSpace->{0,20}]
     {UniversalSpace -> {0, 20}}
Six1 = FuzzySet[{{3,.1},{4,.3},{5,.6},{6,1.0},
  {7,.6},{8,.3},{9,.1}}]
     FuzzySet[{{3, 0.1}, {4, 0.3}, {5, 0.6}, {6, 1.}, {7, 0.6},

        {8, 0.3}, {9, 0.1}}, UniversalSpace -> {0, 20}]
FuzzyPlot[Six1];
```



Solution 2. A second solution would be to use the FuzzyTrapezoid function to create the fuzzy set. For a case such as this, a triangular fuzzy set would probably be better than a trapezoid, so we set the middle two parameters of the FuzzyTrapezoid function to 6.

```
Six2 = FuzzyTrapezoid[2,6,6,10]
                    1       1       3                3       1
      FuzzySet[{{3, -}, {4, -}, {5, -}, {6, 1}, {7, -}, {8, -},
                    4       2       4                4       2

            1
        {9, -}}, UniversalSpace -> {0, 20}]
            4
FuzzyPlot[Six2];
```

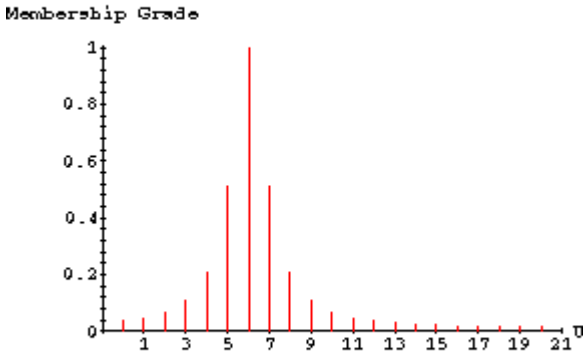


Solution 3. Another solution would be to use a function to create a fuzzy set representing numbers near 6.

```
CloseTo[x_] := 1/(1+(#-x)^2)&
```

We can use this function to create a fuzzy set for numbers near 6.

```
Six3 = CreateFuzzySet[CloseTo[6]]
     FuzzySet[{{0, 0.027027}, {1, 0.0384615}, {2, 0.0588235},

        {3, 0.1}, {4, 0.2}, {5, 0.5}, {6, 1.}, {7, 0.5}, {8, 0.2},

        {9, 0.1}, {10, 0.0588235}, {11, 0.0384615}, {12, 0.027027},

        {13, 0.02}, {14, 0.0153846}, {15, 0.0121951},

        {16, 0.00990099}, {17, 0.00819672}, {18, 0.00689655},

        {19, 0.00588235}, {20, 0.00507614}},

       UniversalSpace -> {0, 20}]
FuzzyPlot[Six3];
```

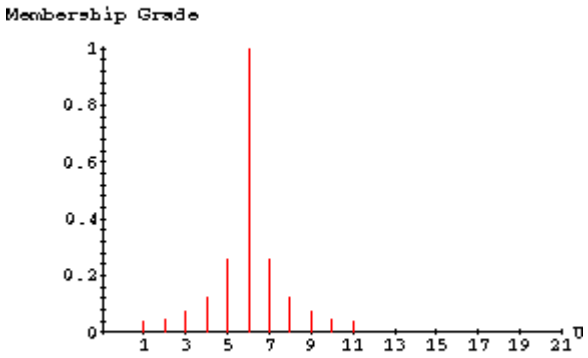


Note that this is a convenient method because the function CloseTo can be called with any integer argument to produce a fuzzy set close to that number.

Solution 4. Still another solution is to use a piecewise function to describe the fuzzy set.

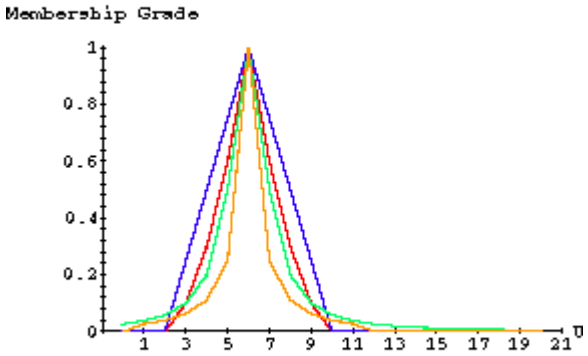
```
NearSix[x_] := Which[x==6, 1,
                     x>6 && x<12, 1/(x-5)^2,
                     x<6 && x>0, 1/(7-x)^2,
                     True, 0]
Six4 = CreateFuzzySet[NearSix]
     FuzzySet[{{1, 0.0277778}, {2, 0.04}, {3, 0.0625},

        {4, 0.111111}, {5, 0.25}, {6, 1.}, {7, 0.25},

        {8, 0.111111}, {9, 0.0625}, {10, 0.04}, {11, 0.0277778}},
```

```
       UniversalSpace -> {0, 20}]
FuzzyPlot[Six4];
```



Now, we can view all four of our fuzzy representations of the number six to see how they compare. We do this by plotting them all on the same graph with the FuzzyPlot function.

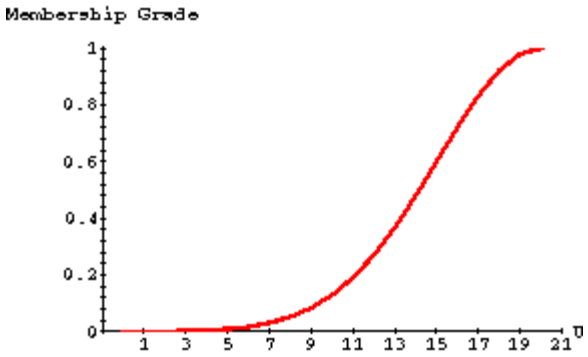
```
FuzzyPlot[Six1,Six2,Six3,Six4,PlotJoined->True];
```



Example 5. Fuzzy Hedges

Problem. Suppose you had already defined a fuzzy set to describe a hot temperature as:

```
Hot = FuzzyGaussian[20,7,UniversalSpace->{0,20}];
FuzzyPlot[Hot,PlotJoined->True];
```



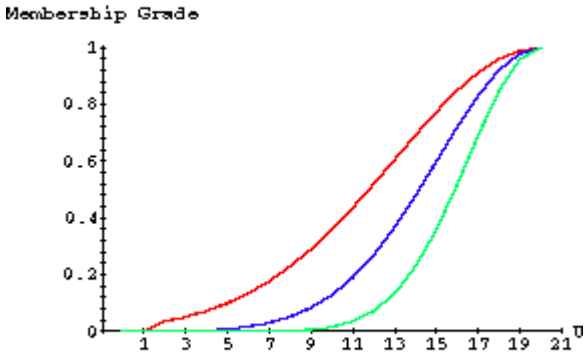
Now, suppose we want to talk about the degree to which something is hot. We need some sort of fuzzy modifier or a hedge to change our fuzzy set. Let us look at how we can accomplish this.

Solution. We can start by defining how a fuzzy set should be modified to represent the hedges 'Very' and 'Fairly .' Two functions in the Fuzzy Logic Pack, Concentrate and Dilate, can be used to define our two hedges.

```
Very := Concentrate
Fairly := Dilate
```

Now we can look at a graph of the fuzzy sets FairlyHot, Hot , and VeryHot.

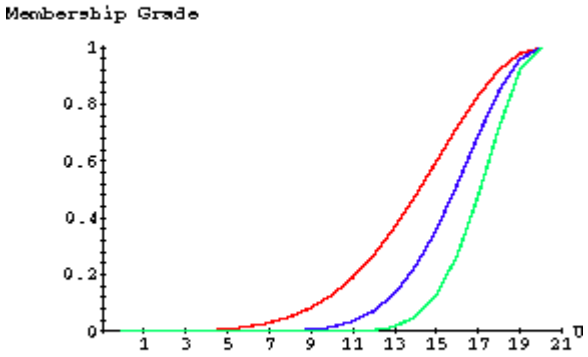
```
FuzzyPlot[Fairly[Hot],Hot,Very[Hot],PlotJoined->True];
```



Note that the FairlyHot membership function is a more general, spread-out fuzzy set. The VeryHot fuzzy set is a more focused, concentrated fuzzy set.

We can also apply more than one modifier to a fuzzy set. For instance, let us compare Hot, VeryHot, and VeryVeryHot.

```
FuzzyPlot[Hot,Very[Hot],Very[Very[Hot]],
  PlotJoined->True];
```



As we might expect, the VeryVeryHot fuzzy set is even more concentrated than the VeryHot fuzzy set.

Example 6. Distance Relation

Problem. Let R be a fuzzy relation between the sets, X={New York City, Paris} and Y={Beijing, New York City, London}, that represents the idea of "very far." In list notation, the relation could be represented as follows [Klir & Folger, 1988].

R(X,Y) = 1/NYC,Beijing + 0/NYC,NYC + 0.6/NYC,London + 0.9/Paris,Beijing + 0.7/Paris,NYC + 0.3/Paris,London

Solution. We can represent this fuzzy relation in Mathematica in the following way. We can start by creating the membership matrix to represent the relation.

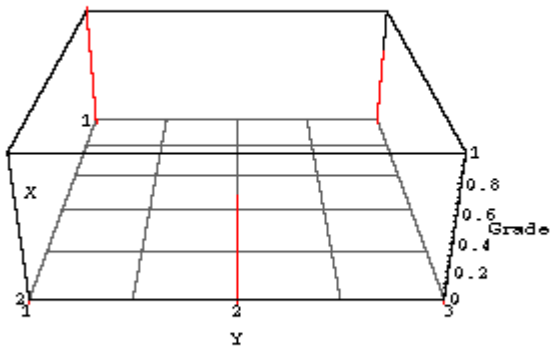
```
DistMat = {{1,0,0.6},
  {0.9,0.7,0.3}}
    {{1, 0, 0.6}, {0.9, 0.7, 0.3}}
```

We need to represent the cities in each set with numbers. For set X, let New York City be 1 and Paris be 2; for set Y, let Beijing be 1, New York City be 2, and London be 3. Now we can create the relation using the FromMembershipMatrix function.

```
DistRel = FromMembershipMatrix[DistMat,{{1,2},{1,3}}]
     FuzzyRelation[{{{1, 1}, 1}, {{1, 2}, 0}, {{1, 3}, 0.6},

        {{2, 1}, 0.9}, {{2, 2}, 0.7}, {{2, 3}, 0.3}},

       UniversalSpace -> {{1, 2}, {1, 3}}]
```

We can plot this relation using the FuzzyPlot3D function. We will use some of Mathematica's Plot3D options to put the graph in a form that lines up with the membership matrix so that you can see the correlation.

```
FuzzyPlot3D[DistRel, AxesLabel->{" X","Y","Grade  "},
  ViewPoint->{2,0,1},AxesEdge->{{-1,-1},{1,-1},{1,1}}];
```




```
ToMembershipMatrix[DistRel]
     1     0     0.6

     0.9   0.7   0.3
```

By customizing the graph, you can get it to match the membership matrix, which makes understanding the fuzzy relation easier.

Example 7. Choosing a Job

Problem. Fuzzy sets can be used to aid in decision making or management. We illustrate this with an example from Klir and Folger [Klir and Folger, 1988]. Given four jobs (Jobs 1, 2, 3, and 4), our task is to choose the job that will give us the highest salary, given the constraints that the job should be interesting and close to our home.

Solution. The first constraint of job interest can be represented with the following fuzzy set.

```
Interest = FuzzySet[{{1,.4},{2,.6},{3,.8},{4,.6}},
              UniversalSpace->{1,4}]
     FuzzySet[{{1, 0.4}, {2, 0.6}, {3, 0.8}, {4, 0.6}},

       UniversalSpace -> {1, 4}]
```

We can see that Job 3 has the highest membership grade, meaning that Job 3 is the most interesting of the Four jobs. Job 1 on the other hand is the least interesting, since it has a membership grade of only 0.4.

We can form a fuzzy set for our second constraint in a similar manner. Here is a fuzzy set used to represent the driving distance to the four jobs.

```
Drive = FuzzySet[{{1,.1},{2,.9},{3,.7},{4,1}},
          UniversalSpace->{1,4}]
     FuzzySet[{{1, 0.1}, {2, 0.9}, {3, 0.7}, {4, 1}},

       UniversalSpace -> {1, 4}]
```

In the fuzzy set above, the membership grades indicate the length of the drive to work. A high membership grade indicates that it is a short drive to work -- a good thing. A small membership grade indicates an undesirable, long drive to work. From the fuzzy set above, we can see that Job 4 is located near our home, while Job 1 is a long way from our home.

Finally, we need to figure in the goal of a good salary. There is no real difference between a constraint and a goal in this problem, so we figure in the worth of the salary the same way we did for the previous constraints. We could use a formula to convert a salary into a membership grade for each job [Klir & Folger, 1988], but to stay with the tradition of our previous constraints, we arbitrarily assign a membership grade to each job based on salary.

```
Salary = FuzzySet[{{1,.875},{2,.7},{3,.5},{4,.2}},
            UniversalSpace->{1,4}]
     FuzzySet[{{1, 0.875}, {2, 0.7}, {3, 0.5}, {4, 0.2}},

       UniversalSpace -> {1, 4}]
```

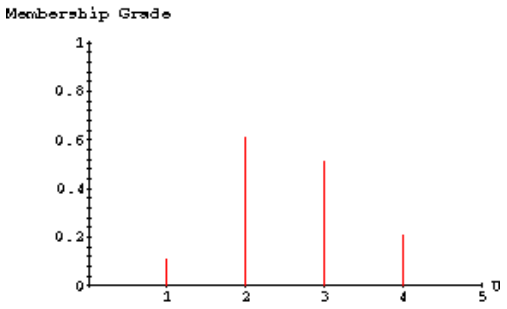
From this fuzzy set, we see that Job 1 pays the highest salary, and Job 4 pays the lowest. Now that all of our criteria is represented as fuzzy sets, we need to decide on a function to make the decision. We will use the standard Intersection to make the fuzzy decision. Applying the

Intersection operation can be thought of as "anding" the constraints and goal to come up with the best overall decision.

```
Decision = Intersection[Interest,Drive,Salary]
     FuzzySet[{{1, 0.1}, {2, 0.6}, {3, 0.5}, {4, 0.2}},

       UniversalSpace -> {1, 4}]
```

We can plot the decision fuzzy set to see the results graphically.

```
FuzzyPlot[Decision];
```



At last, we can look for the maximum membership grade to decide which job best satisfies our goals and constraints. In this example, we see that Job 2 appears to be the best job for us.

There are a number of different ways that the decision in this example could have been made. For example, we could have used a different operator, maybe a product operator, to make our decision; we could have weighted different constraints more heavily than others; or we could have used different functions to arrive at the membership grades. As an exercise, try using a different method and see which job your method selects as the best.

.....