A new GRASS GIS fuzzy inference system for massive data analysis

Jarosław Jasiewicz

Adam Mickiewicz University, Geocology and Geoinformation Institute, Dzielęgowa 27 60-081 Poznań, Poland

ABSTRACT

GIS systems are frequently coupled with fuzzy logic systems implemented in statistical packages. For large GIS data sets including millions or tens of millions of cells, such an approach is relatively time-consuming. For very large data sets there is also an input/output bottleneck between the GIS and external software. The aim of this paper is to present low-level implementation of Mamdani’s fuzzy inference system designed to work with massive GIS data sets, using the GRASS GIS raster data processing engine.

Keywords:
Fuzzy logic
GIS
Data analysis
Inference system
GRASS

1. Introduction

Many spatial phenomena show a degree of uncertainty that cannot be expressed with clear-cut boundaries. In continuous spaces where it is difficult to clearly define boundaries, terms like “close to”, “suitable” and “small” can better express the character of divisions than a simple binary true/false classification (Burrough, 1992, 1996). Additionally, fuzzy systems are useful on areas where data are incomplete or where there are only general rules of system behaviour. Because of these qualities, fuzzy methods have been successfully applied in numerous GIS fields such as data classification (Fisher, 1999; Demicco and Klir, 2004), geomorphology and soil mapping (Burrough, 1989; Burrough et al., 1992; MacMillan et al., 2000; Scull et al., 2003), image classification (see: Petry et al., 2005), decision support systems (McBratney and Odeh, 1997), hydrology and flood risk (Makropoulos et al., 2008), ancient settlement studies and historical data (Borodkin, 1999; Barceló, 2005; Jasiewicz and Hildebrandt-Radke, 2009), and tourism (Ergin et al., 2003).

Mamdani (Mamdani and Assilian, 1975) was among the first to apply fuzzy logic to build control systems based on Zadeh’s (1965, 1973) fuzzy set theory and fuzzy decision system (Zadeh, 1973). There are several implementations of fuzzy inference systems based on Mamdani’s solution in high-level mathematical and statistical programming environments such as Mathematica, MATLAB (MATLAB Fuzzy Logic Toolbox Documentation) or R (Meyer and Hornik, 2009). These systems, however, are designed to be used as control systems or decision support for small data sets. Their application for large GIS data sets comprising millions or tens of millions of cells (numbers of iterations) is time-consuming.

Input/output operations (especially for large data sets) between GIS and external applications are additional bottlenecks. Hence, the full GIS implementation of inference systems capable of obtaining a satisfactory analysis time requires a low-level approach.

The presented fuzzy system for GRASS GIS is a low-level implementation of Zadeh’s (1973) and Mamdani’s fuzzy inference system (Mamdani and Assilian, 1975), which is designed to work with GIS raster databases of any size and type supported by the GRASS GIS system. The GRASS GIS, an open-source GIS environment (Neteler and Mitasova, 2008), was chosen because of its ability to process very large data sets with low memory load. The system consists of three modules: r.fuzzy.system, r.fuzzy.sets and r.fuzzy.logic. The r.fuzzy.system is not required to work with the two additional modules, which have been designed to perform basic fuzzy operations and overlays and to test intermediate results. All three modules are written in ANSI C using GRASS API and can be run on all platforms that support GRASS GIS. Like other GRASS programs, all modules use command line interface and GRASS Tcl/Tk or wxPython GUI’s.

The proposed fuzzy system is not the first implementation of fuzzy logic in a GRASS GIS environment. For the 4.0 version of GRASS GIS, there is an external fuzzy logic system created by François Delclaux in 2000. Unfortunately, it is no longer supported by current versions of GRASS. The system described in this paper is a completely new solution unrelated to GRASS 4.0.

2. Data processing

Classical mathematical variables usually use numbers as values. Linguistic variables use linguistic values (linguistic terms), which
are usually words. In the GIS fuzzy system, linguistic variables refer to the original raster data (map), whereas linguistic values (terms) are equivalent to the fuzzy set created over every given map according to a uniform definition. The inference process is defined by the set of fuzzy rules of the "IF antecedent THEN consequent" type, where the antecedent is the conditional part and the consequent is the conclusion part of the rule. The antecedent consists of one or more fuzzy variables and terms connected by the semantic operators (AND, OR); consequent consists of only one variable and term, for example: "IF distance_to_river is near AND elevation_above_river is low THEN risk is very high". The full fuzzy system comprises input and output data, the fuzzy set definition, a system of rules and an inference machine whose work can be controlled by additional parameters.

2.1. Input data and fuzzy set definition

The input data for calculation are standard raster data stored in the GRASS database. The system uses one or more maps of any type (i.e. integer, float, double precision) and accepts all values appropriate to the given data type. Any cell having a null value is ignored during data processing. The output is one (or optionally more) floating point raster map containing the final result. The analysis involves continuous data processing and does not require intermediate stages. The input information is passed to the system as a text file. The file contains map names used in the analysis (analogue of linguistic variables) and definitions of fuzzy sets (analogue of linguistic values) for every map. The map file also includes one output map definition. To identify the output map in the rule file, it must be named _OUTPUT_. while the output map name is passed as a standard command parameter (see Appendix). The range of all sets in the output map is used to define the universe of the consequent, while the resolution of the universe is also a command parameter.

Membership modelling is very important in fuzzy logic concepts. In the proposed system, modelling of a fuzzy set is limited to a generalised trapezoid but with additional modification possibilities. It is also possible to define one-sided sets with only two points and information as to whether the points define the left or right boundary of the set. The shape of the boundary of the set may be "linear", "s", "g-" or "j-shaped". Additional modifiers are hedge and height. If the hedge parameter is other than from zero, dilatation (negative) or concentration (positive) is applied on the defined fuzzy set as many times as the number in the hedge field. The height parameter defines the maximum membership of the set. Height affects both the maximum membership of the set and its boundaries. This approach allows precise modelling of fuzzy set boundaries without visual feedback. Examples of fuzzy sets based on different definitions are shown in Fig. 1.

2.2. Rule parser

Since the described modules are designed for users to build their own fuzzy inference systems, the most difficult problem is to
predict how many rules and how many variables will be used in
every rule. Similar to the input data, the rule system is passed to the
system as a text file. The text file consists of rules where the
consequent names must be the same as the names of fuzzy sets
defined for the output map in the map file. To store rules, the file
uses the special syntax described in Table 4 and the program
documentation. The text file is processed only once (including
syntax error checking) at the beginning of the program run. Every
rule in the text file is parsed into two complementary FIFO queues,
which store respectively the following tokens: (1) symbols of
values, operators and parenthesis; (2) pointers to the addresses in
the memory with current values. An additional queue keeps
pointers to fuzzy set definitions. The rule parsing process is based
on the idea implemented by Niemann (2010) and uses bottom-up
shift reduce parsing (Aho et al., 2007), where tokens are shifted to
two stacks – operator stack and value stack – and the current
action of the parser is determined by the precedence table. This
solution makes that number of rules, and variables in rules are not
arbitrarily limited. During data processing, the parser refers always
to the same memory locations that store subsequently new data
during processing of cells. Regardless of the size of input data,
during processing all rasters are loaded into memory row by row;
cells of each row are processed sequentially so memory loading
during data processing does not exceed a few Mb, depending on the
number of maps and the number of rules used in the analysis. This
approach allows the processing of massive data sets of any size
supported by GRASS GIS system, even those in excess of the amount
of available memory.

2.3. Inference machine

The process of fuzzy inference in the r.fuzzy.system consists of
five steps that are similar to solutions in other software (Meyer,
2009, MATLAB Fuzzy Toolbox Documentation): (1) fuzzification
of the data; (2) application of fuzzy logic operators in the antecedent
(3) implication from antecedents to consequents; (4) aggregation
of all consequents across rules; (5) defuzzification. The first three

Fig. 2. Fuzzy inference procedure in r.fuzzy.system module.
operations are performed separately for every rule; the fourth operation aggregates into one fuzzy set, and the last operation changes the fuzzy set (which cannot be stored in classical GIS data) into one value to store it in the resulting map (Fig. 2). All of these steps are iteratively repeated for every cell in the GRASS region. The additional parameters for controlling the inference process are supplied as standard GRASS module parameters.

During the fuzzification process, original values are recalculated into membership grades for each fuzzy set associated with the given map. The result is a group of single values [0,1] equal to the number of sets for every map. Application of the fuzzy logic operation is performed from left to right according to the string of operators, variables and values stored in the antecedent section of every rule. If a rule contains an operator of negation, the complement of the original value is taken for further operations. Fuzzy logic operations may use different fuzzy logic families previously defined in the program. There are six popular families available (see: Klement et al., 2000; Fodor, 2004; Bobillo and Straccia, 2009; Meyer and Hornik, 2009 and Table 1 for details). The operation results in a separate group of single values for every antecedent. The number of antecedents is equal to the number of rules. Because every rule has only one final antecedent value, there is an additional option in the program that allows the user to output all the antecedents as a set of maps. These maps use the final output map name as the prefix and the rule name (consequent) as the suffix. The result map contains floating point values of the output universe. The additional category file shows also the membership function of every cell in particular output fuzzy sets; for example:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>none</td>
<td>$0$</td>
</tr>
<tr>
<td>$1$</td>
<td>high</td>
<td>$1$</td>
</tr>
<tr>
<td>$2$</td>
<td>veryhigh</td>
<td>$2$</td>
</tr>
</tbody>
</table>

In the third step, every antecedent value is mapped onto the consequent fuzzy set, resulting in a new consequent fuzzy set. This implication procedure can use either of two options: a minimum, where the result is the minimal value of the antecedent and current consequent for every element, or a product, where the result is a product of these values. The fuzzy set definitions for the consequents are stored in a map file with an output map.

Finally, all consequents are aggregated into a single consequent using the semantic OR operator, and are defuzzified into a single value. There are several defuzzification methods described by Leuwick van and Kerre (1999). The methods used in the r.fuzzy.-system are: centroid (mass centre), bisector (fuzzy set divided into two equal area), Min. of highest, Max. of highest, Mean of highest. The result map contains floating point values of the output universe. The additional category file shows also the membership of every cell in particular output fuzzy sets; for example: high: 0.71 veryhigh: 0.29. These memberships for every cell can be interactively checked by clicking on it.

2.4. Additional output and associated modules

For visual control of the intermediate stages, the program can output a fuzzy set definition for every map and consequents for every rule, as well as aggregated consequents in the form of comma-separated lists of values that can be visualised by any external software.

### Table 2
Performance tests of r.fuzzy.system due to (A) data resolution; (B) output universe resolution; and (C) defuzzification method. The fuzzy logic family has no influence on module performance.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>No. of cells</th>
<th>Time</th>
<th>Width</th>
<th>Height</th>
<th>Time</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A) Module performance due to data resolution (universe resolution = 100; family = Zadeh; defuzzification = centroid).</td>
<td>2800×380</td>
<td>106400</td>
<td>6994950</td>
<td>1388×1399</td>
<td>2790×3758*</td>
<td>6990×9495*</td>
<td></td>
</tr>
<tr>
<td>No. of cells</td>
<td>1.2 s</td>
<td>7 s</td>
<td>27 s</td>
<td>27 s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>1 min 46 s</td>
<td>10 min 55 s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B) Module performance due to universe resolution (data resolution = 1388×1399; family = Zadeh; defuzzification = centroid).</td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td>6</td>
<td>9</td>
<td>16</td>
<td>27</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (s)</td>
<td>27.6</td>
<td>27.1</td>
<td>7.5</td>
<td>7.6</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Original Spearfish location with oversampled data.
Additional modules: the r.fuzzy.set and r.fuzzy.logic are not required to perform the inference process. The module r.fuzzy.set may be useful for testing the result of applying any fuzzy set on the original map, while r.fuzzy.logic allows the analysis of the results of fuzzy operations (OR, AND, NOT) performed on fuzzified data with one of the proposed fuzzy logic families. Of course, these modules may also be used separately with other GIS tools.

3. Performance

The speed performance tests are presented in Table 2. Every test was run after cold restart of the computer. All tests were performed on an AMD X2 250 processor with 4 Gb RAM and 64 bit Ubuntu 9.04 as an operating system, and GRASS GIS version 6.5.svn (6 June 2010). The input data and rule system were the same as in the case study example below. The relationship between number of cells and computation time is linear. On testing machine about 91,000 ± 11,000 cells per second were calculated, regardless of data resolution. Aggregation is the most resource-consuming part of the data processing, hence the universe resolution strongly affects computational time. If high precision is not required, the resolution can be set to the lowest value; however, with several or more rules, low resolution may give highly inaccurate results. The defuzzification does not affect the computational time; the higher speed for the last three methods is due to code optimisation—in “highest” cases, only the highest consequent is aggregated.

The r.fuzzy.system was compared with the fuzzy_inference() function from R-CRAN “Sets” package (Meyer, 2009) at the smallest (280x380 cells) resolution with a simple R script. However, the computation was still continuing after six hours, indicating that there was no comparison between the two approaches.

4. Case study example

The application of the GRASS fuzzy system is presented with a simple inference system prepared to detect the theoretical flood risk on a standard GRASS GIS dataset in Spearfish Country, South Dakota (available at: http://grass.osgeo.org/download/data6.php). From a digital elevation model of resolution 10 m per cell (elevation.10 m map) the flow accumulation, stream network, watercourse distance to streams and the elevation above streams were calculated with the new r.stream toolset (Jasiewicz and Metz, submitted for publication). These data were used as linguistic variables and fuzzy sets ascribed to maps as linguistic terms (see Table 3). The rules system assumes that areas distant from the stream network or lying high above it are free from flood risk, that the low-lying areas or areas with a high flow accumulation have at least a small flood risk and that areas lying near the stream channels have a high flood risk. The rule system is presented in Table 4 and input data in Fig. 3. Three examples with resultant flood risk maps created with the (A) Zadeh fuzzy logic family and the centroid as the defuzzification method; (B) drastic and max_of_highest; and (C) Hamacher and mean_of_highest respectively are presented in Fig. 4. Results show that both family and defuzzification method affect the final result but the general conclusions require more testing on different data sets and rule systems.

5. Conclusion

The proposed fuzzy logic system is a fast, easy-to-use tool that enables the construction of customised fuzzy logic systems without any programming knowledge. For any data stored in raster format, it requires only the definition of the group of fuzzy sets over every map and the preparation of the rule system. Of course, the Table 4

<table>
<thead>
<tr>
<th>Rules file for presented example.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax explanation:</td>
</tr>
<tr>
<td>$ consequent {antecedents}</td>
</tr>
<tr>
<td>$ none {distance=veryfar</td>
</tr>
<tr>
<td>$ low {distance=medium &amp; elevation=high}</td>
</tr>
<tr>
<td>$ moderate {(distance=medium</td>
</tr>
<tr>
<td>$ high {(distance=medium &amp; elevation=low)</td>
</tr>
<tr>
<td>$ veryhigh {distance=medium &amp; elevation=low}</td>
</tr>
</tbody>
</table>

Fig. 3. Source data (fuzzy variables) used in the case study: (A) Elevation with stream network modelled with r.stream (Jasiewicz, Metz, submitted for publication); (B) MFD accumulation map created with r.watershed; (C) watercourse distance to the stream; (D) watercourse elevation above the stream. All scales in meters, except (B) where scale is in log10 of number of cells.

Please cite this article as: Jasiewicz, J., A new GRASS GIS fuzzy inference system for massive data analysis. Computers and Geosciences (2010), doi:10.1016/j.cageo.2010.09.008
Appendix. GRASS commands required to reproduce the example presented in the paper.

```bash
r.watershed -f elevation=elevation.10 m@PERMANENT accumulation=accum
r.mapcalc accum_abs=abs(accum)
r.stream.extract elevation=“elevation.10 m@PERMANENT” threshold=1000 stream_rast=“stream” direction=“dirs”
r.stream.order stream=streams dir=dirs horton=horton
r.mapcalc horton3=if(horton > 2,1,0)
r.stream.distance stream=horton3 dir=dirs dem=elevation.10 m@PERMANENT distance=distance elevation=elevation
r.fuzzy.system maps=“~ /flood.map” rules=“~ /flood.rul” family=“Zadeh” defuz=“centroid” imp=“minimum” res=100 output=“flood”
r.fuzzy.system maps=“~ /flood.map” rules=“~ /flood.rul” family=“Hamacher” defuz=“mean_of_highest” imp=“minimum” res=100 output=“flood”
```

Fig. 4. Resultant flood risk calculated with three different fuzzy logic families and defuzzification methods: (A) Zadeh and centroid; (B) drastic and max_of_highest; (C) Hamacher and mean_of_highest. On the left of every image is a magnified inset of the outlined area.

References


Please cite this article as: Jasiewicz, J., A new GRASS GIS fuzzy inference system for massive data analysis. Computers and Geosciences (2010), doi:10.1016/j.cageo.2010.09.008


