Inline Classification

High Level Requirements

The following are the high level requirements for this design.

- Reduce the use of intermediate data sets when processing the data.
- Provide a single interface to a complete data definition for display, analysis and processing.
- Provide the ability to use the raw data and apply weights or descriptions at runtime.
- Implement the enhancements so that they are backwards compatible.

Requirements

The main requirement is to use the original data sets and add some value-added information when needed to reduce the need for intermediate datasets. This is often just classification for simplification or analysis.

To meet the functional requirements this design must meet the following requirements:

- Reduce or eliminate the need to create intermediate files for solution processes.
- Provide a single source of classification definition (color, weights, descriptions) in one easy to use operation.
- Provide runtime class assignments that are applied when a function is executed.
- Provide the ability to have multiple classification tables assigned to each data layer.
- Provide an easy way to change the assignments for different conditions or solution changes without having to create intermediate data sets again.
- Reduce the number of stored data sets to a single source data set that can be reused repeatedly in different functions and processes with different classes.
- Minimize the impact of introducing the new functionality in the GIS system.
- Minimize the impact on any existing functions that will utilize the new functionality.
- Provide additional functionality to existing functions without changing their core operations.

Classes Overview

Class tables allow a user to define a condition to be used as part of processing and visualizing digital data using a lookup table format. A class table operates like an IF THEN ELSEIF structure where each input value is evaluated to see if it matches a user specified text string or fits into a specified numeric range. If there is no match, then the operation moves on to the next condition statement in the table. The last default action assigns a NULL value to the output if none of the previous conditions are met.

Class Table Definition

All GIS systems already have a class table creation function in the display window that is used to define the visual presentation of selected covers defining ranges or values and assign colours to them with a colour number or an HSL / RGB definition. This function specifies the input data cover and either uses their numeric values or an attribute that holds the values to be classified. There is also functionality that associates the source data with its table definitions.

To implement the class table definition two new columns could be introduced into the editor that allows a user to enter their own user defined numeric values (Wt) and a text description (Des) associated with each range. The Wt column are numerical values entered by users and they are used for any operations involving calculations like statistics. The Des column are strings entered by users and they are used for any operations involving counts.

If there are no values in the second range column then an exact match is made of the entry in the first column. This column should allow both character and numeric values for exact matching.

The advantage of using the color assignment functionality is that:

- The data and conditions are easily displayed with their assigned colors for visual validation of the ranges.
- The existing color assignment functionality can already:
    - Assign ranges to an input data layer.
    - Associate multiple tables to an input data layer.
    - Provide functionality to retrieve a requested table.
- Any changes made to ranges or assignments are easy to see with a visual data representation.

The goal is to present a single interface to users to define the class ranges with their color, weight and description values.

Data Definitions

The ability to reference and use data and definitions in the GIS are important to avoid confusion for both end users and automated processes. Any definition will work as long as it can define all the information required. For this design we will use the data definition defined below.

Data are be defined with a concatenated name that contains the:

- Data layer name,
- Attribute name if it is required,
- Class table name,
- Column in the class table to be used.

The layer name and attribute define the source of the raw data and are separated with a period. The class table named and column define the modification of the return values and are separated with a period. The data definition and the modification definitions are separated with an underbar.

- dataLayerName.Attribute_classTableName.columnName

If we are using a point, vector or raster cover with a class table that return weighted data values, it would look like this:

dataLayerName_classTableName.columnName

Example: Aspect_Heat.Wt

If we are using a point, vector or polygon feature cover with attributes and a class table that returns the data values as descriptions, it would look like this:

dataLayerName.attributeName_classTableName.Des

Example: Soils.SoilType_ErosionWeights.Des

Functional Algorithm Design

The ”inline” operation for the use of class tables would require changes to functions GUIs and their code behind as well as new data definitions and changes to the getData functions in the GIS.

Note: This design assumes that the existing GIS system has getData functionality for all data types that are called to access any data cover values. This includes single value raster, point, line data layers and feature cover attributes with variable or classified attributes.

The backend getData functionality would need to be modified to use the proposed naming convention to define the data and any associated class tables with:

- dataName.attributeName_classTable.columnName for features and
- dataName_classTable.columnName for raw data (points, lines, rasters).

The getData functions would have a process that would parse the input data definition string looking for a period, underbar, period in order to define the data and any runtime transformations.

Note: If no class table is defined then the raw data value is returned. The color column is always used for display. if a columnName is not supplied after a classTableName, the default is to use the weight column for the return.

Note: If the system is designed with a simple string name being sent to the getData functions, then the impact of implementing these can be hidden from the rest of the system or any existing processes that use it, as the string definition proposed here could be parsed inside the getData function. Hopefully this would make the change backwards compatible.

GUI Design

The use of class tables for inline operations would require the addition of a new panel in all the function GUIs in the system. It would have two scrolling selection boxes. One for the class tables associated with the function and one to select the class table column to be used.

When a cover is selected in the Input Cover field its associated class tables are listed in the Class Tables field. The Class table name and column fields default to blank. If the user selects a class table and a column in the Class Table. The class table and column definitions are added to the Input Cover name in the Input Cover field.

As each input layer is selected the Class Table field will only show class tables associated with that cover, so a single class table selector can be used by multiple input layer fields.

Example Operations

The creation of class tables was discussed above with changes to the existing colour assignment functionality in the GIS with the addition of user editable columns in the interface. Just like the colour assignment, each class table is associated with its cover, so a user can select one for that specific cover when needed. An example of the Aspect class table Heat4 Weight is shown below:

Cover:Aspect

Attribute:

Name:Compass4

Column:Wt

RMin /String	RMax	Col RBG/HLS	Wt (Weight)	Des (Description)
0	45	#/#/#	0.1	North
46	135	#/#/#	0.5	East
136	225	#/#/#	1.5	South
226	315	#/#/#	0.8	West
316	360	#/#/#	0.1	North

The class table selector displays the table names associated with the cover in the input cover box. If a table is selected the definition of the input cover is concatenated with an underbar separator.

The Compass4 table is created for the Aspect cover with weighting as the output values. If the input cover has been defined in the function with Aspect_Compass4.Wt the input values for the Aspect cover will be retrieved for each Aspect grid position and run through the Compass 4 table to get their weights which are returned to the function for execution. This results in several useability improvements:

- No need for an intermediate weighting cover as it is applied "inline" in the calculation.
- The Compass4 class table can be reused multiple times.
- Changes to the existing Compass4 ranges, weights and descriptions are easy to apply.
- Different weighting tables can be used for subsequent runs.

Note: The use of inline class tables in the function adds new functionality (conditionals) to the function without changing the core operations or code.

Implementation

The implementation of this proposed functionality would require changes to the:

- Visual colour display table and their interface with two new columns added for user defined values.
- GUI screens for all the functions that have an input data layer field to:
    - Add new panel with a scrolling, select and edit field for the class table name and a column selection field.
    - Specify the complete input data definition with layer, attribute (if needed) and class table name, column (if needed).
    - Apply the class table name to the input definition by displaying the concatenated data definition in the input field after the class table and column selections have been made.
- getData functionality to implement the new:
    - Input data strings and parse out the various definitions.
    - Return value assignments.

Hopefully the processing changes will be straightforward to implement, but the complete solution requires changes to a large number of function's GUIs that would probably have to be included in a product release. The basic file storage, retrieval and usage could be implemented without any GUI changes being introduced. This would allow the use of class tables in batch and modelling processes. The GUI changes could be implemented in a later version for a complete implementation.

Summary

The inline classification functionality should:

- Have the complete definition of classes and their values all in one place.
- Reduce or eliminate many of the intermediate data sets that are used in today's GIS processing.
- Let a user use a single core data set in multiple processes.
- Provide the ability to easily vary the conditions in a process without having to create a new intermediate data cover.
- Make it easier to use functions standalone as it provides the introduction of conditions into the functions execution at runtime without changing the function.
- Enhance existing functionality in GIS systems without impacting their basic operations.

If the system has well-defined getData classes then the implementation of the core inline functionality could be relatively straightforward.

There is a lot of repetitious work implementing the changes in all the functions GUIs to expose the class tables for each data set and creating the resultant definitions of the classes.

Feedback

The concepts and ideas in this design document have been developed on my own and need some constructive criticism from someone with GIS product development experience.

Most importantly has the concept been clearly defined and does it have value to GIS users. I also may have made incorrect assumptions and there could be logic errors in any high level algorithm descriptions or pseudo code examples. I would appreciate any feedback you might have on the concepts and the design.

If this design has merit, I have a couple of other designs I am looking at that might be of interest.

Thanks for reading the document and I look forward to hearing from you.

My contact information is Steven Paine at

stevenhpaine@gmail.com