12B.7

RADAR ECHO CLASSIFIER ALGORITHM DEVELOPMENT USING PYTHON

Joseph VanAndel*
National Center for Atmospheric Research [1]
Boulder, Colorado

1 INTRODUCTION

We have developed a Radar Echo Classifier (REC) algorithm to identify various radar echoes such as precipitation, clear air, and in particular anomalously propagated (AP) ground clutter in NEXRAD radar data. The REC uses fuzzy logic to identify these various echo types. We implemented the REC using the Python language along with C++ extensions. This implementation replaces and extends the functionality of an earlier program, the AP Clutter Analysis Tool (APCAT) (VanAndel, et al., 1999). Our new implementation provides a more productive environment to quickly evaluate and test new algorithms and a very efficient system to process large amounts of data.

2 OVERVIEW OF RADAR ECHO CLASSIFIER

2.1 Anomalous Propagation

Anomalous Propagation (AP) is caused by variations in temperature and moisture in the atmosphere that change the index of refraction. These gradients in the index of refraction can cause the transmitted microwaves from weather radars to “bend”, striking the ground, rather than penetrating the atmosphere. This results in radar echoes from the ground being erroneously interpreted as weather by automatic hydrology programs, causing overestimates of precipitation. Since weather radars are also used to vector aircraft traffic around severe weather, AP clutter can cause unnecessary delays because air traffic controllers may route aircraft around non-existent storms. We are optimizing automatic algorithms to detect AP clutter using “fuzzy logic” (Kosko, 1992, Kessinger, et al, 1999).

2.2 Fuzzy Logic

We use a fuzzy logic algorithm to identify AP clutter in radar data. The first step is to calculate “feature fields” from the reflectivity, velocity, and spectrum width fields of the radar data. Each “feature” is a statistic computed for a given spatial region. Computed statistics include mean, median, standard deviation, and “texture” (mean squared differences over a region).

For each “gate” (spatial location), each feature field is looked up in its “membership function”. The membership function is a mapping from input values to probability. For example, the membership function for mean velocity of AP has a value of 1 for the velocity value of 0, since AP clutter tends to have 0 m/s velocity. The weighted outputs of the membership functions are summed to yield a combined probability output (see Figure 1). (Kessinger, et al., 1999) provides a detailed description of the algorithm.

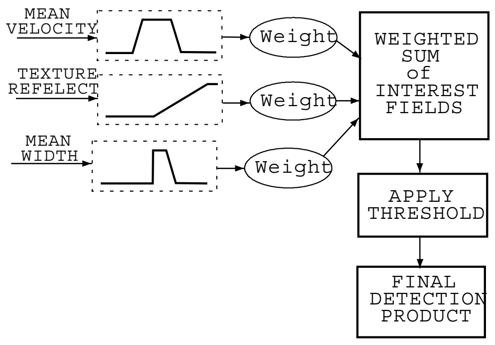

Figure 1 - Fuzzy Logic Computation

2.3 Scoring and Statistics

After computing the Radar Echo Classifier output, the output is scored to evaluate the recognizer performance. The algorithm’s performance is compared to either a scientist’s manual truthing or an objective truth obtained from dual polarization radar data (Kessinger 2001). Statistics are computed and then plotted as shown in Figure 2. Based on these statistics, the membership functions, weights, and thresholds are adjusted to optimize the Radar Echo Classifier.

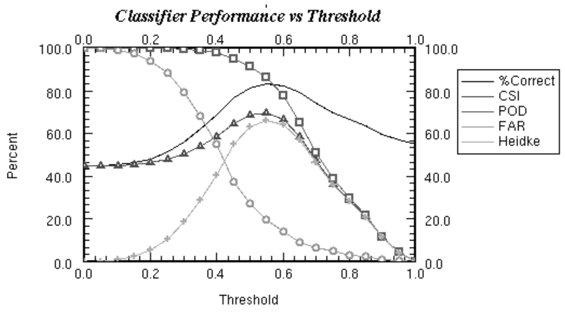

Figure 2 - Performance Statistics

3 WHY DID WE REPLACE OUR EXISTING IMPLEMENTATION?

Previously, we implemented a fuzzy logic recognition algorithm in APCAT (VanAndel 1999). We found several deficiencies with this program that convinced us to write a new program to provide this functionality. First, because of byte-order and alignment issues with our radar data files, we could not easily port APCAT from Solaris to Linux.

*Corresponding author address: Joseph VanAndel, NCAR, P.O. Box 3000, Boulder, CO 80307-3000; email vanandel@ucar.edu

[1] NCAR is sponsored by the National Science Foundation.

Inexpensive Linux workstations are now providing better price-performance than Sun workstations, so we were motivated to build a more portable program. Second, APCAT was slow in processing data because it frequently wrote data files to disk for intermediate storage. Since modern workstations have far more memory than was previously available, the new program could keep entire "sweeps" (360 degrees of radar data) in memory, rather than writing out all the intermediate data to disk. Third, APCAT's user interface and program logic were written in Tool Command Language (Tcl) (Ousterhout, 1994). It became clear that Tcl did not scale well to larger applications because of its lack of object orientation and higher-level data types. Python was a natural choice because it was object-oriented and had superior data types. Finally, APCAT was a single monolithic program that could not easily be attached to other radar processing algorithms. Using Numeric Python (section 4) to implement the REC made it much easier to add new routines. By simply defining new routines that read Numeric Python arrays, it was quite simple to add new algorithms that interface with our existing programs. As a result, we decided to implement our fuzzy logic recognition algorithm using Python.

4 WHAT IS PYTHON?

Python is a very high level interpreted object oriented programming language (van Rossum and Drake 2000) with a rich set of data types. Although Python is easy to learn, it is a very powerful language. Python is freely available, widely used, and well supported on a variety of computers and operating systems. Because Python is interpreted, it is not suitable (by itself) for computationally intensive tasks. However, Python is designed to allow extensions that can perform specialized and computationally intensive tasks. The Numeric Python extension (Ascher 2001) was developed to efficiently perform calculations on large arrays of numbers. Many other extensions are available, such as database interfaces, graphic user interfaces (GUI), web programming, and distributed programming.

5 SOFTWARE ARCHITECTURE

We built the Radar Echo Classifier (REC) software as a set of C++ extensions for Python and a collection of Python scripts. First, we built a set of Python extensions to read NCAR's DORADE format radar files into Numeric Python arrays. Once the radar data was represented as Numeric Python arrays, we could use the standard Numeric Python operators (such as add, subtract, multiply, FFT). We wrote our own Numeric Python extensions to efficiently calculate the "feature fields"(section 2.2). The membership function lookup routine was written in Python using an interpolation routine provided by Numeric Python. Also, we developed scoring and statistics routines that analyzed the performance of the REC. The software architecture is shown in figure 3.

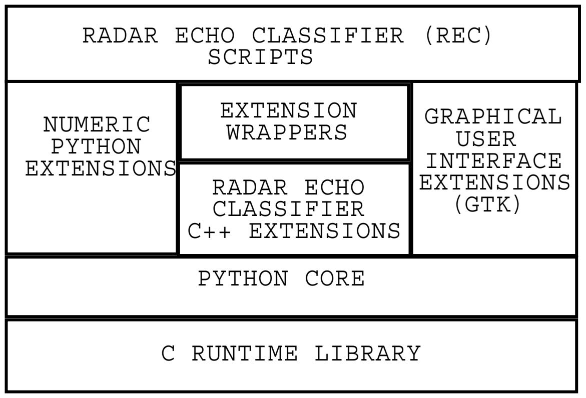
RADAR ECHO CLASSIFIER (REC) SCRIPTS
NUMERIC PYTHON EXTENSIONS
EXTENSION WRAPPERS
RADAR ECHO CLASSIFIER C++ EXTENSIONS
GRAPHICAL USER INTERFACE EXTENSIONS (GTK)
PYTHON CORE
C RUNTIME LIBRARY

Figure 3 – Software Architecture

Because Python is interpreted, it was simple to define our fuzzy logic calculation parameters in a "preference" file that is read when the Radar Echo Classifier starts running. The preference file defines the membership function for each "feature field" along with the weights used to calculate the final result. This implementation makes it very simple for scientists to experiment with new membership functions or weights, without requiring any changes to the Radar Echo Classifier application code.

6 BENEFITS OF USING PYTHON

We found that implementing the Radar Echo Classifier in Python had several benefits. First of all, since Python is interpreted, it was much faster to develop and test new algorithms, compared to the edit, compile, debug cycle of compiled languages. We could easily build small test cases to verify our algorithms. It was much easier to add new features than if we only used a compiled language. Since the computationally intensive algorithms were written as C++ extensions, our classifier could quickly process large amounts of data, even though large portions of our application were written in an interpreted language. We've found that Python is very stable and remarkably bug free, partly because the source is available to everyone, and many other programmers are using it (and contributing their bug fixes.) Since the source is freely available, we've been able to add features to the Numeric Python package itself. Also, since Python is freely available, we can share our work with other researchers, without requiring them to purchase software licenses, as would be required with a commercial software package. When we have needed assistance with Python, we've found the Python development community has quickly responded to our questions.

7 CONCLUSIONS

Numeric Python provided an excellent framework for radar computations. We have been able to take advantage of a large set of existing array manipulation software and have been able to easily implement our own routines within the Numeric Python framework. We would strongly recommend this approach to anyone processing large quantities of numeric data.

8 FUTURE DIRECTIONS

We plan to expand the Radar Echo Classifier by adding a “Confidence Algorithm” based on history, the state of the clutter bypass map, climatology, and terrain data. We intend to add histograms and scatter diagrams to help analyze the statistics of our data. We plan to implement “multi-variable” fuzzy logic, where the membership function used for a particular variable depends on the value of a second variable.

9 ACKNOWLEDGEMENTS

The National Weather Service Radar Operations Center (ROC) in Norman, OK, sponsors this research. We are grateful to the developers of Python and particularly the developers of the Numeric extensions.

10 REFERENCES

Ascher, D., P.F. Dubois, K. Hinsen, J. Hugunin, and T. Oliphant, 2001:*Numerical Python*, Lawrence Livermore National Laboratory. See http://sourceforge.net/projects/numpy

Kessinger, C., S. Ellis, and J. VanAndel, 1999: A Fuzzy Logic Radar Echo Classification Scheme for the WSR-88D, *Preprints,* 29th International Conference on Radar Meteorology, AMS, Montreal, 12- 16 July 1999, 576-579

Kessinger, C., S. Ellis, and J. VanAndel, 2001: NEXRAD Data Quality: The AP Clutter Mitigation Scheme, *Preprints,* 30th International Conference on Radar Meteorology, AMS, Munich, 19-24 July 2001

Kosko, B., 1992*: Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Prentice–Hall, N.J.

Ousterhout, J.K., 1994, *Tcl and the Tk Toolkit*, Addison Wesley

VanAndel, J., C. Kessinger, and D. Ecoff, 1999: APCAT: An AP Clutter Analysis Tool, *Preprints*, 29th Radar Meteor. Conf., AMS, Montreal, 12-16 July 1999, 267-269

van Rossum, G., F.L. Drake, *Python Tutorial,* October 16, 2000 http://www.python.org/doc/current/tut/tut.html (checked 2001/4/2)