INCREASING GEOCOMPUTATIONAL INTEROPERABILITY: TOWARDS A STANDARD GEOCOMPUTATION API

Yunping Liu, Mark Gahegan and James Macgill

GeoVISTA Center, Department of Geography,
Pennsylvania State University,
University Park, PA 16802, USA
Tel: +1-814-865-1666
Email: yzl120@psu.edu

BIOGRAPHY:

Yunping Liu is a Ph.D student from the Department of Geography, Pennsylvania State University. His research interests include GeoComputation, geo-spatial data mining and knowledge dis covery, information visualization, and exploratory data analysis.

INTRODUCTION:

In response to the increasing volume and complexity of geospatial data and problems, the primary concern of GeoComputation is to enrich geography with a toolbox of methods to model and analyze a range of highly complex, often non-deterministic problems (Gahegan 2000). Over the past several years, the collection of GeoComputation tools has grown in an unbounded and diverse manner to address a wide range of techniques and applic ation domains from various perspectives inspired by the related fields such as statistics, pattern recognition and artificial intelligence. While plurality and inclusiveness have been regarded as the strength of GeoComputation, they have limited the succes sful deployment of GeoComputation tools to real world problems because a number of interoperability issues are raised by the diverse and highly specialized expertise required to understand the proposed tools and use them appropriately and also the heterogeneity introduced into the system by their complex and diverse functionality. Three specific problems arise. First, how do we make the tools easily accessible to end users and third party application developers by hiding the technical and implementation details? Second, if the targeting problem requires the integration of tools from different parties, how do we make them work together seamlessly given that they are often developed in isolation with no thought to their eventual integration into some larger system. Third, how do we make the tools from different parties "hot swappable" so that users can experiment freely with a wide variety of methods without having to continually re-engineer the supporting data and control interfaces. To address these interoperability issues, we argue that the GeoComputation community needs a standard operating environment that embraces a wide variety of GeoComputation data, models, methods, training and validation techniques and results. Such a standard environment will be specified in the format of Application Programming Interface (API). An API, a term from computer science community, refers to a set of routines, protocols, and tools for building software applications. Currently, there is no widely agreed upon, standard API for GeoComputation. In this paper we propose to specify a pure Java API to facilitate development of GeoComputation-enabled applications. The Java GeoComputation (JGC) API allows java based GeoComputation tools to be engineered to a single uniform interface that can be understood by a wide variety of client application developers and end users. Similarly, GeoComputation applications can be coded against a single API that is independent of the underlying host system. The JGC specification implements much of the new Java Data Mining API (JDM, JSR73), defined by a cross-disciplinary alliance of developers, sponsored by both Sun and Oracle and now beginning to be taken up by a number of open-source computational projects (Hornick et al. 2004). The benefits of JGC specification to the GeoComputation community are obvious:

1. Increases the interoperability of GeoComputation tools, facilitates the sharing of GeoComputation tools both within and outside the GeoComputation community, and reduces the cost and potential risk of deploying GeoComputation tools to real world applications.
2. Contributes to the development of scientific standards in GeoComputation and therefore may help to reduce some of the reluctance among the quantitative analysis community to adopt GeoComputational tools (Couclelis 1998).
3. Provides better support for the automation of analysis and modeling functions, such as finding the optimal configuration parameters for a machine learning method, and helps create end-user friendly GeoComputation technology (Openshaw 2000).
4. Promotes the sharing of GeoComputation technologies, services and information resources in our increasingly distributed and mobile society.

CURRENT DEVELOPMENTS

The proposed JGC architecture consists of three logical components, the API, the GeoComputation Engine (GCE), and the metadata repository (MR). The API is the end-user-visible interface that allows access to services provided by the GCE. The API shields the GeoComputation user from details about the actual implementation and supporting components of the GCE. The GCE provides the infrastructure that offers GeoComputation services to users through the API defined above. The GCE can be implemented as a server in which case it is called a GeoComputation Server. The third component is the metadata repository (MR) that is used by the GCE to make various objects (such as datasets, configuration parameters, results) persistent. The metadata repository stores these objects so that they can be used by the GCE to support its operations. The metadata repository may exist as a flat file system or a relational database. The three logical components can be implemented as one physical system or in a distributed environment. Figure 1 shows three possible architectures for a JGC implementation. A JGC implementer may add additional utilities and management interfaces to enhance its JGC implementation, but these additional components are not part of the JGC specification. Similarly, a JGC implementer may choose to implement a subset of the JGC specification to support only portions relevant to his problem domain.

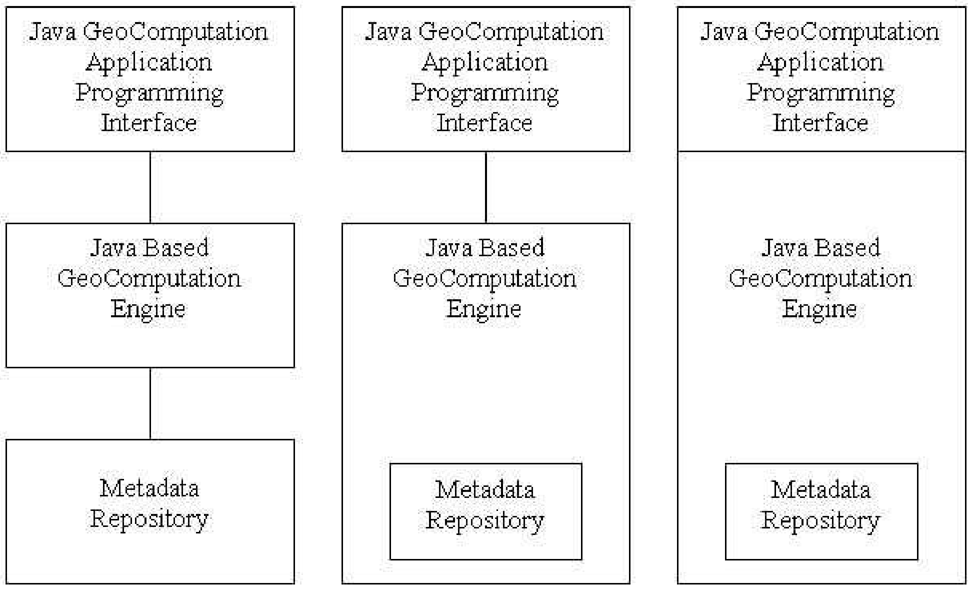
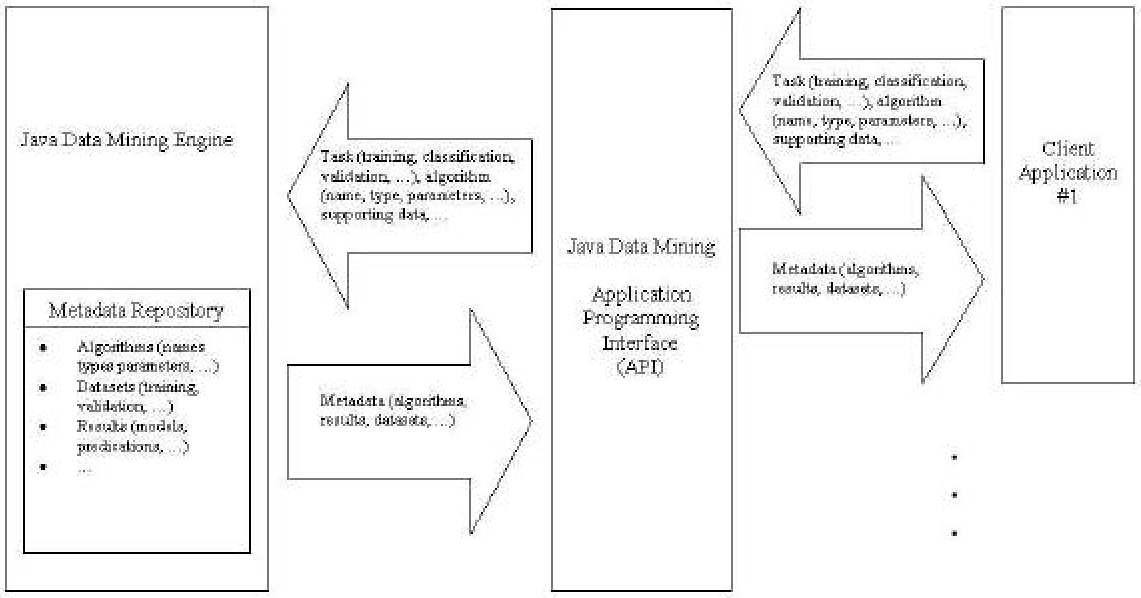

Java GeoComputation Application Programming Interface
Java Based GeoComputation Engine
Metadata Repository
Java GeoComputation Application Programming Interface
Java Based GeoComputation Engine
Metadata Repository
Java GeoComputation Application Programming Interface
Java Based GeoComputation Engine
Metadata Repository

Figure 1. Architecture configuration options for implementing a Java GeoComputation specification

We demonstrate the utility of the JGC API to support the unsupervised and supervised classification and clustering of geospatial data, via a suite of classification and clustering methods that includes quantiles, equal intervals, standard deviation, k-means, k nearest neighbor, maximum likelihood, linear regression, linear discriminant analysis and quadratic discriminant analysis. While we implemented the Data Mining API and Data Mining Engine (DME) as one physical system, it can be easily converted to a distributed architecture such as the client-server architecture, where the Data Mining Engine is implemented as a dedicated server that provides Data Mining functionalities to client applications through the Data Mining API. In a large scale distributed architecture such as the grid architecture, the Data Mining API and Data Mining Engine can be implemented as a set of services in the Middleware Layer, which intelligently directs user applications in the higher level Application Layer to appropriate computing, storage or other resources in the lower level Resources Layer. These services facilitate a flexible and efficient delivery of Data Mining functionalities depending on the computational, data or other constraints.

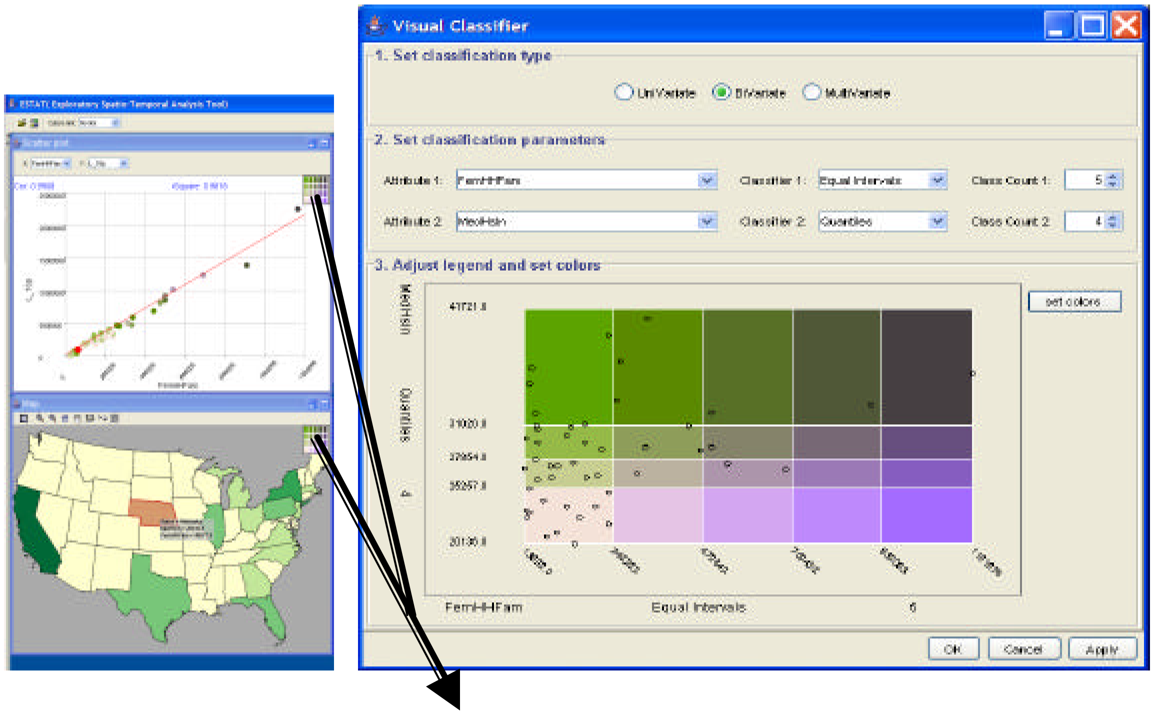
To highlight the benefits of using our API, we will show the ease with which our classification and clustering tools can be accessed, configured, swapped, and connected to other supporting tools such as map visualization. The scheme in figure 2 shows how client applications interact with the Data Mining API.



Java Data Mining Engine
Metadata Repository
Algorithms (names types parameters, …)
Datasets (training, validation, …)
Results (models, predications, …)
…
Task (training, classification, validation, …), algorithm (name, type, parameters, …), supporting data, …
Metadata (algorithms, results, datasets, …)
Java Data Mining Application Programming Interface (API)
Task (training, classification, validation, …), algorithm (name, type, parameters, …), supporting data, …
Metadata (algorithms, results, datasets, …)
Client Application #1

Figure 2. Interaction between user applications and the Java Data mining API

We experimented with linking a real world application developed at the GeoVISTA center, the Exploratory Spatio-Temporal Analysis Tool (ESTAT, a collection of GeoVISTA *Studio* components), to our Data Mining API. Figure 3 shows the user can access various classification tools packaged in our Data Mining API from applications such as scatter plot and map visualization through a graphical user interface (GUI).



1. Set classification type
( ) UniVariate [✓ BiVariate] ( ) MultiVariate

2. Set classification parameters

Attribute 1:	FemHHFam	Classifier 1:	Equal Intervals	Class Count 1:	5
Attribute 2:	MedHsIn	Classifier 2:	Quantiles	Class Count 2:	4

3. Adjust legend and set colors

Clicking on the color legend brings up the classification interface

Figure 3. ESTAT users can access classification tools provided by the java Data Mining API

The GUI shown in figure 3 is a tentative one and needs further development work to fully exploit the flexibility and other advantages offered by our Data Mining API. More details about the proposed JGC specification, our case study and application demonstration will be discussed in the full paper.

REFERENCES

Couclelis, H. (1998). GeoComputation in context, in P. Longley, S. Brooks, R. McDonnell and B. Macmillan (Eds), GeoComputation: A Primer, 17-30, Chichester: Wiley.

Gahegan, M. (2000). What is GeoComputation? A history and outline. http://www.geocomputation.org/what.html

Hornick, M. et al. (2004). JSR73: Java Data Mining API. http://www.jcp.org/en/jsr/detail?id=73

Openshaw, S. (2000). GeoComputation research agendas and futures, in S. Openshaw and R. J. Abrahart (Eds), GeoComputation, 382, London: Taylor & Francis.