A novel visual modeling system for time series forecast: application to the domain of hydrology

Mutao Huang and Yong Tian

ABSTRACT

Accurate and reliable forecasts of key hydrological variables such as stream flow are of importance due to their profound impacts on real world water resources applications. Data-driven methods have proven their applicability to modeling complex and non-linear hydrological processes. This paper presents a novel visual modeling system that has been developed to overcome the problems involved in implementation of data-driven models for hydrological forecasts using conventional programming languages: problems such as the effort and skill needed to program the models, the lack of reusability of existing models, and the lack of shared tools to perform tedious tasks such as preprocessing data. The system provides an integrated visual modeling environment within which users are able to graphically design and verify specific forecasting models for particular problems without writing code. A set of popular data-driven models are offered by the system. Plug-in models created by wrapping existing code are also allowed to run within the system due to the system's open architecture. The system's feasibility and capability is demonstrated through a case study of forecasting 1-day ahead flow in a river basin located in China. The encouraging simulation results show that the system can simplify the process of implementing hydrological forecast.

Key words | artificial neural networks, data-driven model, extreme learning machine, hydrology forecast, support vector machines, visual modeling

Mutao Huang
Yong Tian (corresponding author)
School of Hydropower and Information Engineering,
Huazhong University of Science & Technology,
Wuhan 430074,
Hubei,
China
E-mail: *hmt1973@sina.com*; *ytian.world@gmail.com*

NOMENCLATURE

i	index of input variable
x_i	the ith input variable
t	current time
Δt	lead time
$y(t+\Delta t)$	model output at time $t+\Delta t$
n_i	maximum lag for the ith input variable
k	index of training instance
n	dimension of input vector
$\boldsymbol{x}_k$	the kth input vector
y_k	desired output with respect to $\boldsymbol{x}_k$
K	total number of training instances
P	total number of input variables
G	training data set
Φ	mapping function
$e(t)$	unknown mapping error
Q_e, Q_s	river flow variables
R_j	rainfall variable
j	index of rainfall gauge station
$M_{\max}$	maximum number of hidden neurons for ELM

doi: 10.2166/hydro.2012.158

INTRODUCTION

Forecasting hydrological variables is required to provide basic information for solving water planning and management problems. Over the past decades, much work has been devoted to develop models for modeling hydrological phenomenon. These efforts resulted in a wide variety of models. The existing models may be roughly classified into two primary groups: conceptual or physically based models and data-driven models. The physically based models are of importance in the understanding of hydrological processes, but they often require complicated input data as well as rigid boundary and initial conditions. The higher data requirement becomes a serious barrier for the application of the physically based models, especially in developing countries where the availability of spatial and hydrological data of required quality is rather limited.

In recent years, data-driven models are gaining increasing interest from hydrologists. Their attractiveness comes from that they directly establish mappings between stimuli and response of a hydrological process rather than attempting to reach understanding of internal structure of the physical process (Lorrai & Sechi 1995). The applicability of data-driven models only relies on the availability of recorded time series of environmental observations data. Therefore, employment of data-driven modeling in hydrological forecasts became popular as a result of the growing availability of the data. They were used to forecast real time flood, simulate rainfall-runoff procedure, infill missing data and predict water quality. Most of the existing data-driven models were based upon the methods of computational intelligence and machine learning (Solomatine & Ostfeld 2008). Artificial neural networks (ANNs), model trees, fuzzy-rule based systems, and support vector machines (SVMs) are the mostly utilized techniques for implementing these models.

Despite significant improvements made in enhancing hydrological forecasts using data-driven models, applying these models to real world problems is a huge challenge posed on water resource managers and engineers. On one hand, implementing the model algorithms by writing computer programs often requires comprehensive technical experience and coding skills. Although many software packages such as Matlab allow the reuse of existing data-driven modules or libraries, these modules are standard and they may not be directly used to produce desired results. As a result, considerable efforts are still needed to tune them to specialized applications. Visual modeling technology could be leveraged for simplifying model construction process. It provides necessary abilities to help users accomplish modeling exercises while hiding complexity of the underlying code by providing easy-to-use interfaces. However, tools that enable hydrologists to perform a data-driven modeling exercise from start to finish in a visual environment have not yet emerged. On the other hand, the process of applying data-driven models involves other important steps, such as determining suitable model inputs, building and maintaining a database to store and manage observations data, data formatting and cleansing, data visualization and summarization, and the use of modeling experiences to evaluate the empirical regularities. Performing these tasks requires the modelers to have enough expert knowledge and modeling skills such as databases, statistical analysis, and machine learning algorithms. However, water resource managers and engineers are often not ready to be equipped with these skills. Thus, data-driven modeling approaches are not easy for them to use.

Reuse of existing data-driven models is another challenge that has received little attention in hydrology modeling community. The importance and benefit of model exchange and reuse have been highlighted by the environmental modeling community. Utilization of existing work is much more efficient and potentially more robust when the existing methodology is already proven (Holzworth *et al.* 2010). To facilitate model reuse, there are a growing number of efforts devoted to develop standards and software tools. The US Geological Survey (USGS) Modular Modeling System (MMS) provides a research and operational framework to support the development and integration of a wide variety of hydrologic and ecosystem models (Rizzoli *et al.* 1998; Leavesley *et al.* 2000). Jeton (2001) utilized the Precipitation-Runoff Modeling System (PRMS) incorporated into the MMS to generate streamflow forecasts. The Object Modeling System developed by the US Department of Agriculture is a Java-based modeling framework that consists of a library of science, control, and database modules and a means to assemble the selected modules into an application-specific modeling package (David *et al.* 2002; Ahuja *et al.* 2005). The Open Modeling Interface and Environment (OpenMI) (Moore & Tindall 2005) defines a set of standard interfaces that provide a mechanism by which physical and socio-economic process models can be linked to each other, to other data sources and to a variety of tools at run-time. Although these systems and standards are generally applicable, they are mainly designed to provide unified platforms for running and integrating physically based models. To the best knowledge of the authors of this paper, none of the existing tools or frameworks supports reuse of data-driven models in the hydrology domain.

In a data-driven modeling exercise, sufficient time series of point observations, such as those made at a discharge gauge or a stationary weather station, are required to calibrate the model parameters. However, many applications suffer from the diversity and heterogeneity issues resulting from the observations data. Generally, the data from different providers are stored in different formats (e.g. Excel, database, text, etc.) that are different syntactically (e.g. file types, file formats, and data structure) and semantically (e.g. variable names, and units) from one data source to the next (Horsburgh *et al.* 2009). Therefore, there is a fundamental need for methods to efficiently organize and utilize observational data. The methods should be both capable of handling the diversity and heterogeneity of the data and simple enough to use for the modelers.

All of the above challenges indicate a need for a new system that allows the integration of existing and future data-driven models into a common, and flexible environment. To this end, this paper focuses on the design and development of a visual modeling system with the purpose of providing an interactive environment for simplifying the modeling process of hydrological forecast using data-driven models. The system integrates a visual modeling designer with a set of built-in data-driven components and a variety of auxiliary utilities to support graphical design of forecast model structure and to handle some of the tedious tasks associated with time series, such as consistency-checking, transformation, generation of training data set, etc. The goal is to relieve the modelers from the complexity of underlying code concerns, so that they can focus on more scientific tasks such as analysis, decision-making, and interpretation of modeling results.

The main contributions of this paper are as follows. (1) Design and develop a hydrological modeling system that eases the application of data-driven model for time series forecast. The solution streamlines all the modeling tasks from input data management to visual presentation of the final results. The system may fill a real void in hydrological modeling environments. (2) Define a set of standard interfaces for data-driven modeling. The interfaces provide a unified way for constructing and running data-driven models and enable to avoid duplication of coding efforts by allowing reuse of existing software modules or libraries.

DATA-DRIVEN MODELING FOR HYDROLOGICAL FORECASTING

Essence of data-driven modeling

When modeling hydrological time series, the future output $y(t+\Delta t)$ (e.g. runoff) is often assumed to be related to current and past inputs $x_i\ (t{-}n_i)$

$$y(t+\Delta t) = \Phi(x_i(t), x_i(t-1), x_i(t-2),\ \ldots, x_i(t-n_i) + e(t)) \tag{1}$$

here Φ is the unknown non-linear mapping function; $i = 1, 2, \ldots, P$; P is the number of input variables; t is current time; Δt is lead time; n_i is the maximum lag (also called memory length) for the ith input variable, $e(t)$ is the unknown mapping error to be minimized. If concurrent value for the input variable x_i is available, in other words, $x_i\ (t+\Delta t)$ is known, the concurrent value could be included in the input vector. In hydrology, the values of x_i can be causal variables such as rainfall, temperature, previous flows, water levels, meteorological data, and so on. The values of y can be hydrological responses such as runoff, stream-flow, and others (Govindaraju & Artific 2000).

In general, the data-driven modeling process includes training of the model on sample sets and then using the trained model to forecast (or predict) the behavior at subsequent times (Basheer & Hajmeer 2000). If there are P input time series and one output time series, a training set could be built through synthesizing these time series. The training set is expressed as

$$\boldsymbol{G} = [(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2),\ \ldots, (\boldsymbol{x}_K, y_K)] \tag{2}$$

where the duple $(\boldsymbol{x}_k, y_k)$ is an instance of the training set; $k = 1, 2, \ldots, K$; K is the total number of the training instances; the n-dimensional input vector $\boldsymbol{x}_k = [x_1, x_2, \ldots, x_n]^k$; y_k is the desired output with respect to $\boldsymbol{x}_k$. The process of learning data behaviors of $\boldsymbol{G}$ and constructing empirically the function Φ is called training.

POPULAR DATA-DRIVEN MODELS FOR HYDROLOGICAL FORECASTING

Artificial neural networks

ANNs have been successfully used in hydrology-related areas such as rainfall-runoff modeling (Dibike & Solomatine 2001), stream-flow forecasting (Sivakumar *et al.* 2002; Gopakumar *et al.* 2007), ground-water modeling (Banerjee *et al.* 2011), water quality, precipitation forecasting (Chau *et al.* 2010), and so on. The Multi Layer Perceptron (MLP) network is a typical example of an ANN. Previous experiments (Narendra & Parthasarathy 1990) have shown that MLP trained with a standard backpropagation (BP) algorithm (Fausett 1994) can be used effectively to model complex nonlinear processes. The BP algorithm is perhaps the most popular algorithm for training ANNs. However, the common BP algorithm has several shortcomings (Basheer & Hajmeer 2000). First, the learning process is time-consuming; second, it may be trapped at a local minimum on the error surface. Many variations of the BP algorithm have been proposed in order to improve its performance. In this work, the strategy of adaptive learning rate is adopted to accelerate learning speed, and the technique of jitter is employed to add small random noise to the weights of all synapses at regular intervals during training in order to prevent the network from settling at a local minimum.

Support vector machines

A SVM (Vapnik 1998) is a group of advanced machine learning algorithms that are based on recent advances in statistical learning theory. SVMs have seen increased usage in hydrological forecast application. They were used for flood management in the prediction of river water flows and stages (Dibike *et al.* 2001; Liong & Sivapragasam 2002; Yu *et al.* 2006), rainfall-runoff modeling (Bray & Han 2004), time series forecasting (Pai *et al.* 2010). SVM implements Vapnik's structural risk minimization (SRM) principle. It seeks to minimize the generalization error, i.e. true error on unseen examples, instead of the empirical error based on empirical risk minimization (ERM) in other neural networks. In the literature, support vector regression (SVR) was used to describe regression with SVM. More mathematical details about SVR can be found in several references (Vapnik 1999; Smola & Schölkopf 2004; Hong *et al.* 2011). Determining appropriate values of model parameters is important for the successful application of SVR. Cherkassky & Ma (2004) give an excellent review of the selection of parameters for SVR.

Extreme learning machine

Extreme learning machine (ELM) is an emerging learning algorithm proposed by Huang *et al.* (2006). Recently, ELM techniques have received considerable attention in computational intelligence and machine learning communities, in both theoretic study and applications. ELM has been successfully applied to a number of real world applications (Choi *et al.* 2008; Lan *et al.* 2010; Martínez-Martínez *et al.* 2011), showing good potential for resolving regression and classification problems with an extremely fast learning speed. ELM is based on the single-hidden layer feed-forward network (SLFN) architecture with additive hidden nodes or radial basis function (RBF) hidden nodes. ELM randomly assigns the hidden nodes' parameters instead of adaptively tuning them. The hidden layer maps the input space into a new space using a fixed nonlinear transformation. The output space performs a linear combination on the new space. The only adaptive parameters being solved are the output weights, which can be determined by means of the least-squares method. The preliminary ELM is a batch learning algorithm with a fixed network structure. Huang & Chen (2008) extended the preliminary ELM from the SLFNs with additive or RBF hidden nodes to generalized SLFNs with a wide variety of hidden nodes. However, the issue regarding design of ELM network architecture remains open.

SYSTEM DESIGN AND IMPLEMENTATION

Requirements for the system

After analyzing common characteristics and key procedures of data-driven modeling approaches, we identify the key requirements the system has to meet:

- Provide a user-friendly interface and isolate the users from the complexity of underlying codes. For expert users, the system should provide support for accepting user-developed components to extend the system's capabilities.
- The ability to cope with the diversity of various data sources and to overcome syntactic and semantic heterogeneities that exist between the sources.
- Support for graphical representation of model structure, wherein the user is able to arbitrarily design and modify the structure, explore and edit the model parameters, and test the model performance with visual response.
- Support for choices of alternative models and methods, whenever possible. This enables the user to investigate the performance of alternative models, and find out which of the candidate models is indeed appropriate, thus overcoming the disadvantage arising from systems with only one option.
- Provide shared tools to perform important but tedious tasks such as determination of input variables, building training sets, normalization of data and so on.
- Support for persistence of user-defined model structure and parameters to a file for future use without repeating the modeling steps.

Conceptual design

Given the above requirements, the development of the system started with conceptual design. To cope with the diversity of data sources, relational database management systems (RDBMS) were employed to assemble and achieve observation records, taking advantage of the high performance of RDBMS for querying and managing data. Additionally, the issue of syntactic heterogeneity could be addressed by leveraging RDBMS. Since the system was designed to be independent from the underplaying database, a standard database schema was needed. The Observations Data Model (ODM) (Horsburgh *et al.* 2008) developed by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) was adopted to fulfill this requirement. ODM provides a consistent format for the storage and manipulation of data across different relational databases (e.g. Oracle, SQL Server, Access, MySQL).

In order to make components of the system independent and reusable, a technique called 'design-by-interface' was adopted (Lowy 2005). The core idea behind the technique is to clearly specify the functionalities a component offers and to separate the way the component's functionalities are used from the details of how the components is implemented. For example, when the science of a component needs to fetch data from elsewhere, it calls an interface to get the data. The component does not need to know how the data are obtained or even where the data have come from, only that valid values are returned when requested. This separation provides two immediate benefits. First, it is easy to develop an extensible system by employing a plug-in mechanism. Communications between the host system and its plug-ins are done through these well-defined interfaces. The second benefit is the ability to switch models without affecting the operation of the host system. The host system is not concerned with a specific model, just the common mechanism to communicate with the models. Based on the concept of 'design-by-interface', we designed a suit of standard interfaces aimed to facilitate reuse and linking of various components. The remainder of this subsection describes the ODM and the standard interfaces in detail.

The ODM data model

At present, many standards for describing and sharing environmental observations information are available, e.g. ODM, Observations & Measurements (O&M) schema developed by the Open Geospatial Consortium (OGC), Environmental Sampling, Analysis, and Results Data Standards proposed by the Environmental Data Standards Council (Environmental Data Standards Council 2006), etc. In this work, ODM was selected for several reasons. First, ODM is targeted to point observations, especially to hydrologic data; second, in contrast with the other standards, ODM not only defines the data elements required to describe observations but also provides the format for storage in a relational database; and finally, a variety of software tools have been developed for discovering and retrieving hydrologic and climate data (e.g. HydroDesktop, HydroExcel, HydroSeek) and for assisting with the process of loading data into an ODM database (e.g. ODM tools).

Figure 1 depicts the major part of the ODM data model used in this work. Tables that store data sources, data collection methods, data qualifiers and other information are not presented here for simplification. Readers are referred to Tarboton *et al.* (2008) for the complete ODM design specifications and data dictionary. In Figure 1, the primary table *DataValues* stores the numeric values for observations and links (foreign keys) to all of the data value level attributes. The *Sites* table provides information about monitoring sites. The *Variables* table lists the full descriptive information about what variables have been measured. The *Units* table gives the units name (e.g. Milligrams Per Liter) and units type (e.g. Length, Time, Mass) associated with the variable.

To maintain consistency and to resolve the semantic heterogeneity resulting from the use of synonyms, the approach of controlled vocabularies (CV) has been used in ODM. In Figure 1, the *VariableNameCV*, *DataTypeCV* and *GeneralCategoryCV* tables contain CV for variable names (e.g. 'Temperature', 'Discharge'), data types (e.g. 'Average', 'Minimum', 'Maximum') and general categories (e.g. 'Hydrology', 'Water Quality'), respectively. ODM imposes CV on some fields within the data model. For example, *VariableName* field in the *Variables* table must reference valid terms from the *VariableNameCV* table.

The *SeriesCatalog* table is particular because all information in it is summarized from other tables. It is based on the concept of data series, which is an organizing principle within ODM. Each row in the table represents a distinct data series that consists of all of the data values associated with a unique site, variable, method, source, and quality control level combination. This table is mainly used to accelerate data query. Most common data queries (e.g. what variables have been measured at a specific site, and what data are available for the site) can be performed by querying this table instead of querying the entire *DataValues* table, which may be overly complicated and therefore time-consuming. This entire table is programmatically derived and is updated every time data are added to the database.

The standard interfaces for data-driven modeling

Figure 2 shows the diagram of the standard interfaces for data-driven modeling. The interfaces may be grouped into three categories according to their purposes.

Definition of generic data-driven model structure

An essential part of standardizing the data-driven modeling is schematic representation of the model structure. In general, a forecasting model is comprised of a set of input data series, an output data series and a data-driven computation engine (e.g. ANN, SVM, etc.). The

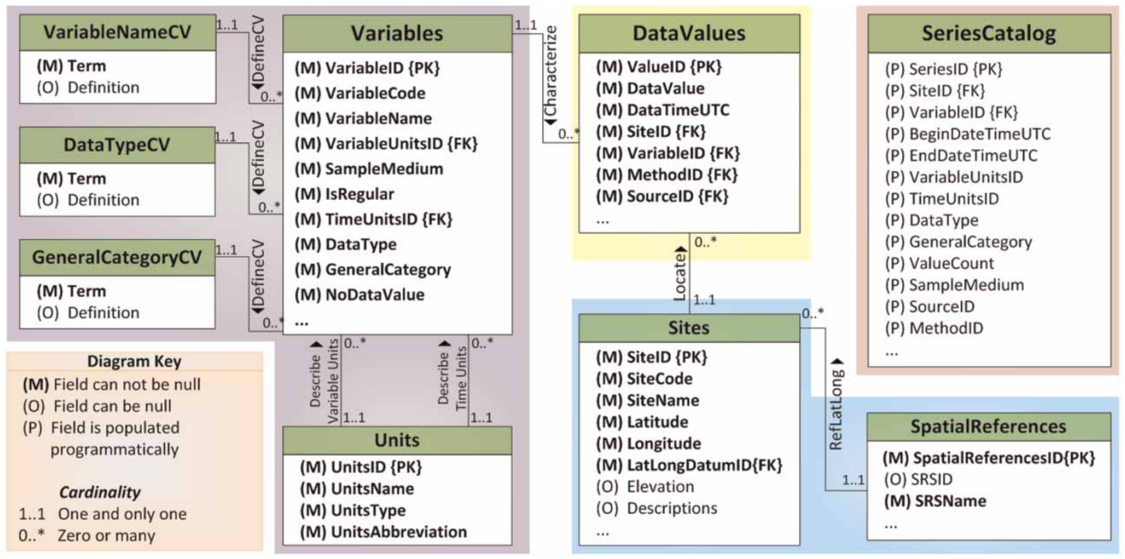

Figure 1 | Logic design of the relational model for point observations data.

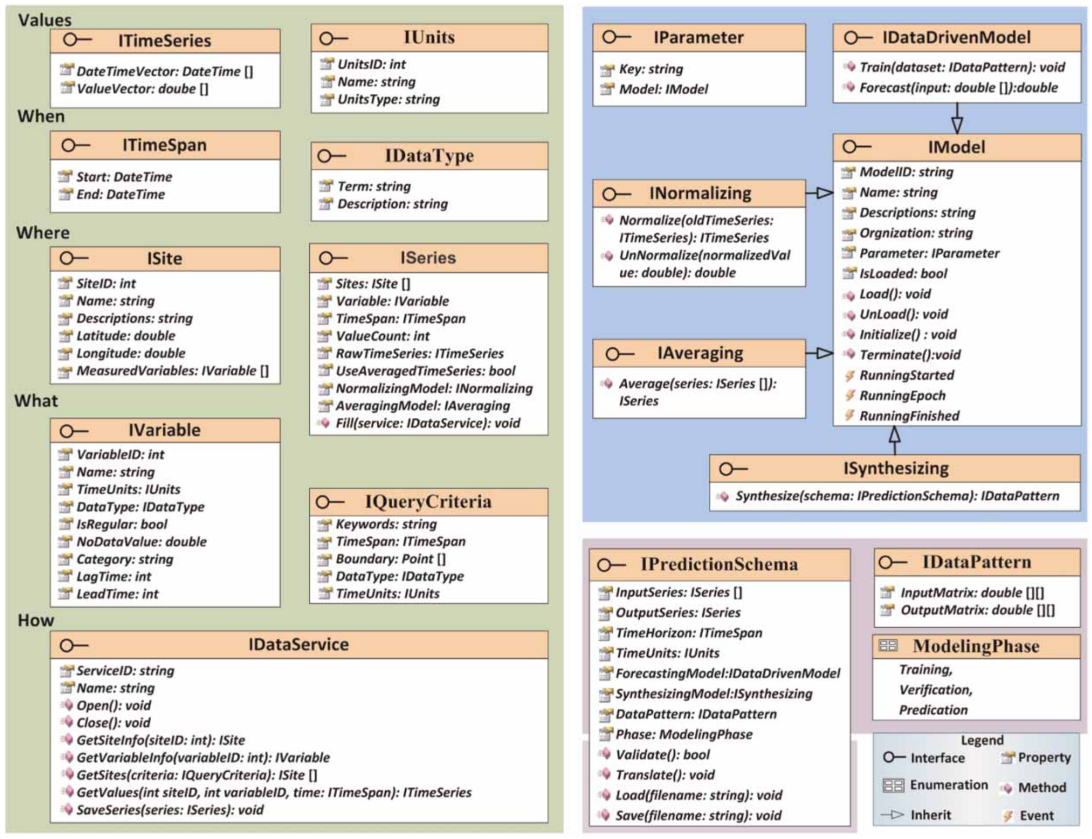

Figure 2 | Diagram of the standard interfaces for data-driven modeling.

IPredictionSchema interface is defined to represent this general structure. It tells what data-driven model is used, what observations sites and variables involved, and how data series are retrieved and manipulated. Both input and output data series are expressed by instances of *ISeries*. The data-driven model is indicated by the *ForecastingModel* property. The method used to build the training set is specified through the *SynthesizingModel* property. The input–output data patterns of the training or forecasting set is represented by the *IDataPattern* interface. The time span over which the model is applied is set through the *TimeHorizon* property. Other global controlling parameters, i.e. time basis for modeling (e.g. day, week, month, etc.) and current modeling phase (training, verification, prediction), are determined via the *TimeUnitsID* and *Phase* property respectively.

The *validate* method of *IPredictionSchema* is typically called by the graphic user interface (GUI) at configuration time when elements of schema have been added, removed or modified. The *Save* method is responsible for saving schema elements, layouts and settings to local files, while the *Load* method is in charge of loading the saved files to reproduce the model structure. *IPredictionSchema* provides a unified and model-independent way to describe a forecasting model structure, it allows different data-driven models to share the same data patterns and thus enables to seamlessly switch to alternative models.

Definition of data structure

In order for the system to manipulate the observations data at an abstract level, a set of interfaces corresponding to ODM data model have been defined. These include the definitions of how time, site, variable and the data themselves are described, and how the data are provided. Time in the system is defined by the *ITimeSpan* interface, which represents a period from the start to end date and time. An individual site is represented by the *ISite* interface. *ISite* contains coordinates (i.e. latitude and longitude properties) of the site's location in a standard geo-reference system (e.g. WGS84). Details of the geo-reference system are recorded by *ISpatialReferences*. The *IVariable* interface gives meta data about a variable, including variable name, time units (i.e. minute, day, week, etc.), data value units, etc. Both the time units and the data value units are represented by the *IUnits* interface. Data type of the variable is indicated by *IDataType*. The lag time for an input variable and lead time for an output variable are specified through the *LagTime* and *LeadTime* property, respectively. It should be noted that the *IVariable* itself does not know if it is an input or an output variable. The role of a variable is determined by *IPredictionSchema*.

Hydrologic time series consisting of data points spaced at uniform time intervals is represented by the *ITimeSeries* interface. *ITimeSeries* has a matrix-like structure with two columns, one column holds a vector of sorted date and time values, and the other holds a vector of numerical values corresponding to the sorted data and time vector. At represent, only time series with regularly spaced data is supported.

ISeries is one of the core interfaces in the design. The combination of variable, site(s), and associated time series is represented by this interface. It may describe a distinct raw data series available in the ODM database or a synthesized data series that are derived from multiple raw ODM data series. As described above, an ODM data series is a set of observation values measured at a specific site for a variable. However, within this system the concept of the data series is slightly different from that of ODM. In hydrological modeling, the average value over an area for some point observation variable (e.g. rainfall) is often needed to be calculated from observations made at different sites because that variable represents only point sampling of the areal distribution. To meet this condition, the *Sites* property of *ISeries* allows multiple sites to be carried with respect to a specific variable, and the *AveragingModel* property enables the binding of a particular model to perform the averaging calculation.

ISeries is mainly used to cache observations data in memory. After setting a time span, calling *Fill* method would retrieve time series associated the given variable-site pair via the given parameter of *IDataService* which provides access to the ODM database. If the *UseAveragedTimeSeries* property is set to 'true' and the *Sites* property indeed holds multiple sites, all the time series corresponding to each of the site-variable pair would be retrieved from the database and then averaged using the given *IAveraging* model when calling the *Fill* method. The *RawTimeSeries* property of *ISeries* holds the retrieved or averaged time series.

The *IDataService* is intended to be a generic wrapper for the components that provide access to specific RDBMS. Low-level details of interacting with the underlying database are hidden by this interface. It is used to retrieve data from the database and to reconcile changes to that data back to the database. *IDataService* exposes a set of methods that allow the querying and retrieval of information about sites and variables (by *GetSiteInfo*, *GetVariableInfo*, or *GetSites*) as well as to drill down time series of interest (by *GetValues*) within the local ODM database. In particular, the *GetSites* method returns all the sites that match a given query criteria. The query criteria represented by *IQueryCriteria* consists of time span, keywords, geographic boundary, data type and time units. The geographic boundary composed of an array of points may be a polygon or a bounding box. It is worthwhile that several methods defined by *IDataService* are analogous to that defined by WaterOneFlow web service. Nevertheless, the aim of *IDataService* is to provide generic means for searching and fetching data in local ODM databases that may be built upon different RDBMS, while the main objective of WaterOneFlow is to define standards for delivering data over the Internet. Apparently, the scope of *IDataService* and that of WaterOneFlow are quite different.

Definition of generic models

Data-driven modeling involves various models with different purposes and different implementations. It is necessary to define an interface that provides generic access to these models. The *IModel* is therefore defined for this purpose. It captures common attributes and behaviors of all the models. *IModel* contains a section of properties which tell the model's unique identification, descriptions, and organization or individuals who developed the model. The *Load* and *UnLoad* methods of *IModel* are typically called by the host system to dynamically load or unload a model. After instantiation of the model-object by its constructor, the

Initialize method is called to populate the object with specific information represented by the *IParameter* interface. Furthermore, the *Initialize* method enables the model to prepare itself if needed, e.g. instantiate the computation engine, allocate memory, etc. After calling this method, the model can be inspected for its meta data. All the parameters needed to be configured are encompassed by an instance of the *IParameter*. Calling *Terminate* will force the model to stop running.

In fact, *IModel* is an abstract interface. It does not distinguish between models with different purposes. As a result, the host system is unable to determine which model should be invoked to perform a particular task during run-time merely using this interface. Thus, interfaces that explicitly describe models' functionalities need to be defined. To this end, four types of the most important and generic models have been identified. These include models that implement data-driven algorithms, models that perform normalization calculations, models that provide strategies for averaging multiple data series, and models determining how to build training sets. Four interfaces, i.e. *IDataDrivenModel*, *INormalizing*, *IAveraging* and *ISynthesizing*, are then defined to expose respective behaviors of the four types of models. Note that all the four interfaces are inherited from *IModel*.

IDataDrivenModel provides a unified entry point to the computation engines of different data-driven models. The data-driven model is calibrated by calling the *Train* method. This method requires a training data set as input to perform the calibration. After calibration, given an input vector, a forecasted value with respect to the input vector would be produced by calling the *Forecast* method. The data sets used for training the model are synthesized from distinct time series related to every input and output series in the model structure. Since the time series is accessible via the *ISeries*, it is easy to build the training set with desired style using a specific approach. Here, the approach of populating the training set is represented by the *ISynthesizing* interface. *INormalizing* and *IAveraging* could be implemented to provide personal models for normalization and averaging calculations.

SYSTEM IMPLEMENTATION

According to the conceptual design described above, a visual modeling system was developed using C# language based on Microsoft .NET Framework (Version 4.0). The Microsoft .NET Framework is a software framework for building and deploying Windows applications and services. C# is the preferred language of choice for developing .NET Framework applications. The .NET Framework in combination with C# offers an ideal environment for developing the presented system due to the following reasons. First, .NET Framework enables managed code to call functions exported from an unmanaged dynamic link library (DLL) via the platform invoked technology. This capability allows the system to interact with existing models written in unmanaged code. Second, .NET Framework together with C# contains a collection of powerful libraries for building expertise applications, especially for developing GUI applications. Third, the powerful reflection mechanism provided by C# enables the implementation of a plug-in architecture easily. Finally, many new features introduced into C# can be leveraged to develop data-intensive applications, e.g. the Language-Integrated Query (LINQ) technology greatly facilitates querying and updating data from different types of data sources.

Figure 3 illustrates the architecture and major components of the system. The architecture uses three-layer approaches common to modern software tools, with a base layer handling data management, a middle layer providing core libraries and basic building blocks, and a user interface layer dealing with user interactions. The base layer contains one or multiple ODM databases that store historic data and instantaneous data if possible. The base layer also provides a set of data adaptors, with each adaptor providing access to a specific RDBMS. All the adaptors implement the *IDataService* interface, thus they act as a bridge between the databases and the data model used within the system. ADO.NET components contained in the.NET Framework were leveraged to develop these adaptors. As long as the data template used by the databases is kept static, the system enables a switch to another database without touching the underlying codes.

The middle layer plays a central role in the architecture. It is responsible for coordinating individual tasks and controlling the simulation logic. It comprises: (1) a diagram engine that provides core functionality for flexible model selection and schema translation; (2) a plug-in loader that allows a dynamic load .NET library assembly in the form of DLL. The assembly should implement any of the interfaces

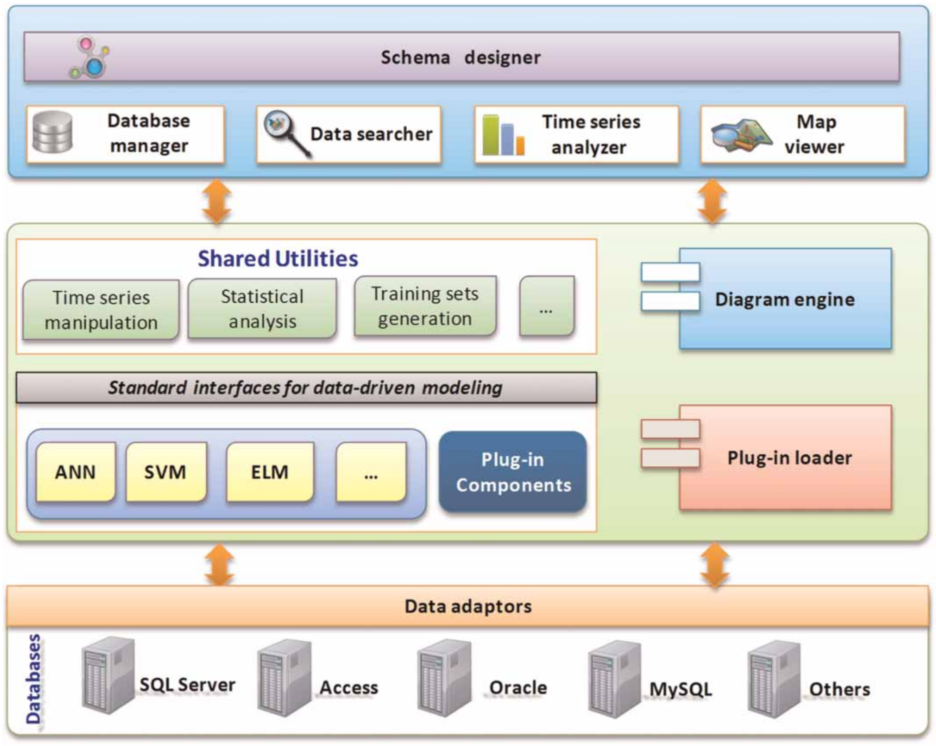

Figure 3 | System architecture and main components.

inherited from the *IModel* interface. When loading such an assembly, the loader looks through the assembly to introspect classes that support the compatible interfaces. Each of the classes may provide a computation engine for a particular model. The model's information could be inspected via a reflection mechanism and then be presented to the user whenever needed, helping the user select the proper model. Methods of the classes could be raised by the host system during runtime, allowing the system to communicate with the plug-in models to take advantage of their designed functionalities. Existing models developed using other programming languages such as VB, and C++ can be migrated to comply with the system through a wrapping pattern. The wrapping means that the model developer creates a C# wrapping class implementing the desired interface. This class is in charge of internal communication with the developer's model engine. After compiling the wrapping class into a .NET assembly, the assembly can be loaded by the plug-in loader; (3) several built-in components that implement the data-driven methods described in previous section; and (4) a set of shared utilities used to manipulate time series, generate training data sets, perform statistical analysis, etc. The middle layer together with the data adaptors in the base layer could be regarded as the system core.

The user interface layer provides an integrated visual modeling environment. Figure 4 shows the main GUI of the system. Several key elements and utilities of the GUI are described below:

1. The schema designer is the core of the GUI. With the designer, the user is able to build and configure schematic representation of the model structure merely through mouse click and drag-and-drop operations. The user can modify the schema easily, e.g. switch from one type of data-driven model to the other, add or delete input variables, specify parameters, etc. The designer has, among other things, the usual features such as moving, or copying elements and the other custom settings. Visualization of large schema is facilitated by zooming.
2. The database manager is responsible for assisting with loading data encoded in other file formats (e.g. .TXT, .CSV, .XML, .XLS, etc.) into the ODM database. It also provides a number of utilities to support manipulation

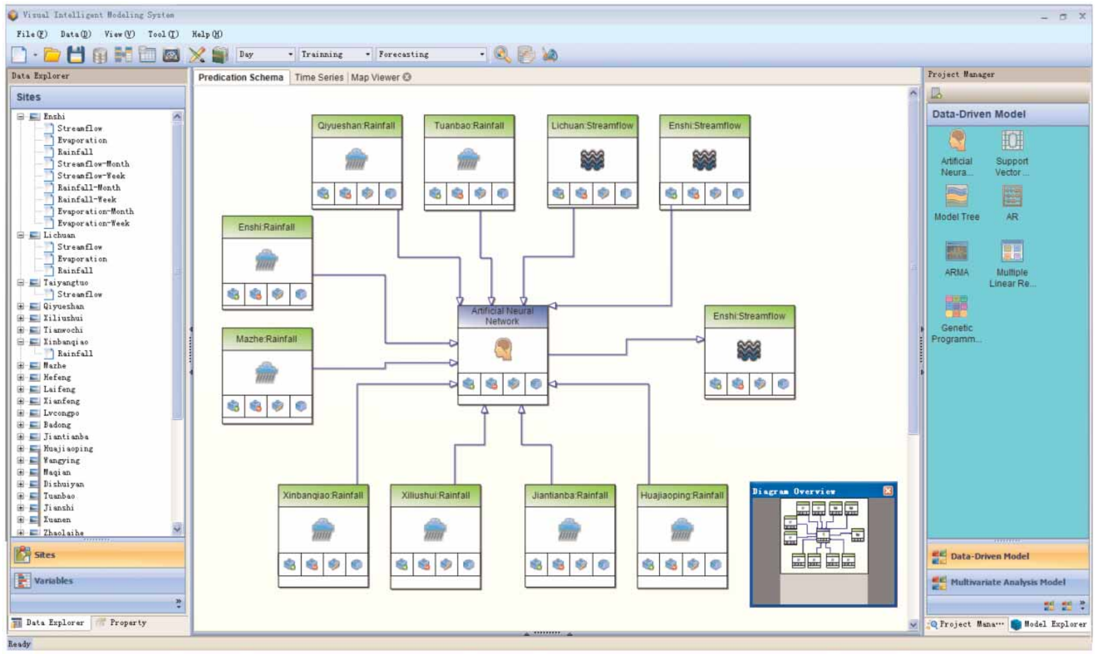

Figure 4 | The main graphic user interface of the system.

of time series, e.g. edit, view, and visualize time series. In addition, new time series could be derived from raw observations data and saved into the database using the manager, e.g. an estimate of mean weekly discharge from the seven discharge values at 1-day intervals.

3. The map viewer provides a set of lightweight geographic information system (GIS) functionalities. It enables the display of geographic locations of sites on a base map (e.g. World Imagery, World Topology map, etc.). The viewer also supports visualization and query of ESRI shapefile data sets. Shapefile representing surface water elements such as watershed or rivers could be visualized and overlayed on the base map, helping the user better understand the spatial context of the hydrological system being studied.
4. The data searcher allows the user to query data series within local ODM database. It contains several utilities to help the user specify query criteria, e.g. an auto-complete textbox that automatically suggests keywords based on the user inputs, a selection tool that allows the user to directly draw geographic boundary on the base map, etc. Here, the supported keywords are targeted to variable names. The query is carried out by searching through the *SeriesCatalog* table. Sites where desired data series are available are displayed on the map viewer through point symbols. Classification of the found data series is presented in the form of a tree view.
5. The time series analyzer uses a collection of systematic approaches to extract information about the characteristics of time series. Approaches to time series analysis include estimating statistical parameters (e.g. mean, variance and others), computing correlation and performing multivariate analysis. The analyzer enables to perform both auto-correlation and cross-correlation analysis. Auto-correlation is the correlation of a time series with itself. The analyzer provides the autocorrelation function (ACF) for calculating serial correlation coefficients (and their standard errors) for consecutive lags in a specified range of lags (e.g. 1–10), and provides the partial autocorrelation function (PACF) for estimating the autoregressive (AR) model order of a time series. Cross-correlation measures the similarity between two time series. The analyzer enables the estimation of the degree to which two time series are correlated. The analyzer also provides methods for analyzing multivariate time series, such as covariance matrix, and principal component analysis (PCA). Correlation methods and multivariate analysis provide analytical techniques for selecting appropriate inputs to a forecasting model. They could be used to recognize the relative

importance of each of the candidate input variables, determine the significant model inputs and specify appropriate lag times for each input variable.

CASE STUDY

In order to verify feasibility and effectiveness of the modeling system, a case study aimed at modeling daily flow in a river basin was conducted. Three data-driven methods provided by the system were used to develop different forecasting models. The fitness of these models was evaluated against the same data sets from the same study area. Comparison of these models was made by means of several commonly accepted performance indices and by comparing the simulated values with the measured values.

Study area and data used

The study area selected is the Qingjiang River basin located in Hubei province, China. The basin is situated between $29^\circ 33' \sim 30^\circ 50'$ E and $108^\circ 35' \sim 111^\circ 35'$ N, and covers an area of approximately 16,700 km^2 with a mainstream length of 423 km. Topography of the basin is hilly with elevation varying from 36 to 2,203 m. The temperature of the basin varies between mean minima of 3.5° (in January) to a mean maxima of 25.5° (in July). The basin is rich in water and hydropower resources. The annual runoff is in excess of 14.1 billion m^3 and the mean annual precipitation is about 1,460 mm. A major portion of the precipitation is received during the monsoon seasons from April to September.

Daily river flow data from two river flow stations, Enshi and Lichuan, and daily rainfall data from eight rainfall gauge stations, were collected for the case study. The available data covered a period of 11 years from 1974 to 1984. The geographic locations of the 10 stations are visualized on the map viewer (see Figure 5). The name of each station is displayed in a popup window that attaches the station's point symbol. The study area is highlighted using a polygon with aqua color, and main streams of the basin are rendered using blue poly-lines.

Implementation of forecasting models

As can be seen in Figure 5, all the stations are distributed on the upstream area of the basin, where the Enshi station is the

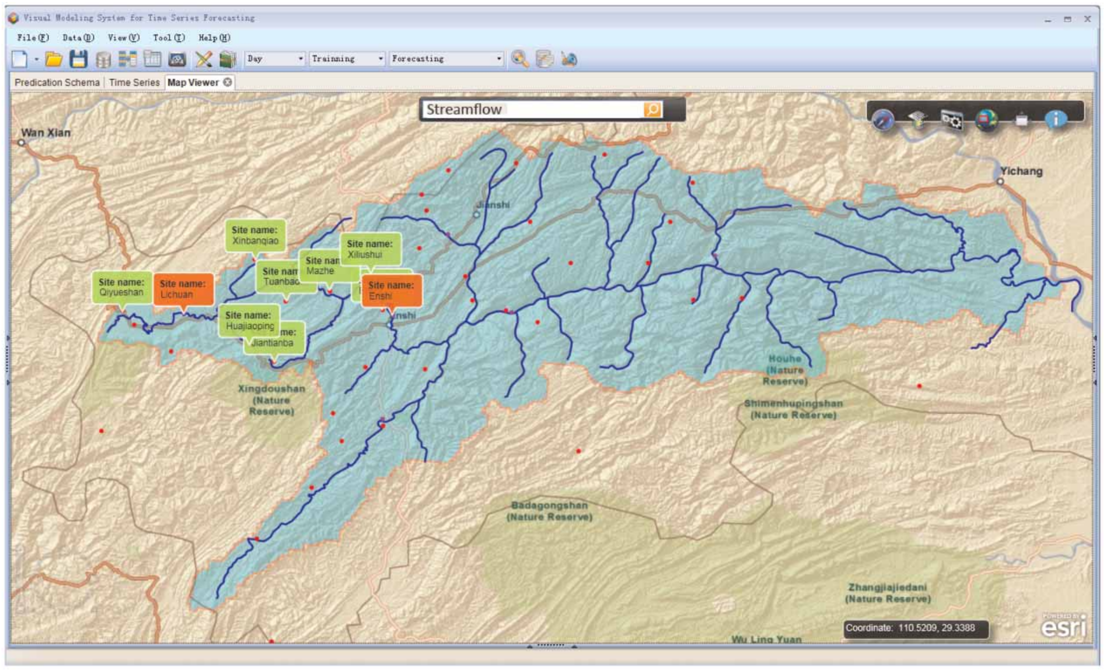

Figure 5 | Map viewer of the system on which observational stations used in the case study are displayed.

outlet of this sub-area. Therefore, a forecasting of daily river flow at Enshi station was modeled. Three forecasting models were constructed using ANN, SVR and ELM, respectively. For the same basis of comparison, the same training, verification and test data sets were used for these three models. The first 7 years (1974–1980) of the available data were used for model training, the following 2 years (1981–1982) were taken as the verification data set and the remaining 2 years (1983–1984) were used for testing forecasting capabilities.

The development of the models started with choices of input variables and determination of lead time for the model output and lag time for each input variable. Rivers in the study area usually showed a fast response to rainfall, therefore, lead time for the output variable was set to 1 day. Systematically determining both the lags and the forecasting model parameters at the same time is prohibitively computationally burdensome, as demonstrated by Bray & Han (2004). Therefore, the lags for the input variables were initially determined using the time series analyzer.

The lags for all the rainfall variables and the flow variable at Lichuan station were determined based on cross-correlation analysis. The analysis results suggested incorporating the past rainfall values up to 3 days' lag and the past flow value up to 4 days' lag in input vector. The lag for the river flow variable at Enshi station could not be similarly determined, instead, PACF analysis was employed. The time series of river flow at Enshi station can be identified as an AR(4) model, so the lag for this variable was set to 4. Therefore, the general structure shared by the three forecasting models has the form

$$Q_e(t+1) = \Phi[Q_e(t), Q_e(t-1), \ldots, Q_e(t-3), R_j(t), R_j(t-1), R_j(t-2), Q_s(t), Q_s(t-1), \ldots, Q_s(t-3)] \qquad (3)$$

where $Q_e(t+1)$ is the river flow to be forecasted at Enshi station for the time $t+1$; $Q_e(t–l)$ and $Q_s(t–l)$ are the inputs of past river flow for the time $t–l$ at Enshi and Lichuan stations respectively ($l = 0, 1, 2, 3$); $R_j(t–r)$ is the input of past rainfall at the jth rainfall gauge station ($j = 1, 2,\ldots,8$; $r = 0, 1, 2$). In Figure 4, the prediction schema corresponding to Equation (3) using the ANN component is displayed. Since the two input variables of rainfall and river flow have different units and different ranges of magnitude, a linearly normalization method was selected to scale all the inputs and outputs in the training and verification data sets to the range of 0.1 to 0.9.

The first set of experiments was made to train the ANN model with BP algorithm. The sigmoid activation function was used for both the hidden and output nodes. Other parameters related to the ANN model were set as follows: initial learning rate was 0.1, jitter epoch (the interval at which jitter is performed) was 50, jitter noise limit (the maximum absolute limit to the random noise added while jitter) was 0.3, and maximum training iteration was 2,000. With this configuration, by training the network with hidden layer nodes varying between 30 and 60, the best model performance was obtained with a minimum of 42 hidden neurons. With the optimum ANN structure consisting of 32 input layer neurons, one hidden layer with 42 neurons and one output layer neuron, Nash–Sutcliffe efficiencies of 90.1 and 86.7% were obtained for the training and verification period, respectively.

The aim of the second set of experiments was to calibrate the SVR model parameters. After trying different combinations of kernel functions (i.e. linear function, RBF and polynomial function) and other related parameter values, the best model performances pertaining to the training and verification data sets were achieved. The RBF was used as the kernel function, gamma value in the RBF was equal to 0.03125. With this configuration, the SVR model achieved Nash–Sutcliffe efficiencies of 93.5 and 89.1% for the training and verification period, respectively.

The third set of experiments was made to investigate the ELM model. The ELM algorithm has two parameters to be determined: activation function of the hidden layer and maximum number $M_{\max}$ of hidden neurons. By varying combinations of the activation function and $M_{\max}$, suitable parameters for the best performing ELM model was identified. In this investigation, sigmoid was used as the activation function, and $M_{\max}$ was assigned to 69.

Performances of the three forecasting models in the training and verification periods are given in Table 1. It can be seen that all the models have satisfactory performance in these two periods. However, the SVR model outperformed the ANN and the ELM model, because it obtained the best R, and NSE statistics in both periods. It is also observed that the ANN needed a relatively longer

Table 1 | The comparison of ANN, SVR, ELM model performances for the training, verification and testing period

	Training				Verification			Testing		
Model	***R***	**RMSE**	**NSE**	**Time**	***R***	**RMSE**	**NSE**	***R***	**RMSE**	**NSE**
ANN	0.950	51.183	0.901	479	0.943	74.223	0.867	0.954	72.39	0.876
SVR	**0.967**	**41.454**	**0.935**	51.7	**0.946**	67.166	**0.891**	**0.955**	**62.921**	**0.907**
ELM	0.956	49.368	0.914	**0.125**	0.942	**63.778**	0.870	0.945	72.435	0.889

time to tune the model parameters and train the network while the ELM ran with an extremely fast training speed.

RESULTS AND DISCUSSION

In order to evaluate the generalization (or forecasting) ability of the trained models, the three models were applied to predict 1-day ahead river flow for the testing period 1982–1984. Table 1 presents a comparison of the forecasting performance of the three models. The overall performance of the models are found acceptable based on the high correlation efficiency. From Table 1 it is obvious to see that the ANN model shows inferior results compared with the other two models while the SVR model holds the best performance. The forecasted river flows produced by the three models versus the measured river flows in the testing period are presented graphically using curves and scatter plotting as shown in Figure 6. The following can be summarized from the figure:

1. It can be seen from the scatter plots that there is relatively good agreement between the simulated and observed hydrographs for all the three models. Thus, it is practically possible to develop daily river flow forecasting models using these data-driven approaches. However, the scatter plots also show that there are discrepancies in matching some of the peak events, where the events may be under predicted or over predicted values. In other words, all the models encounter the generalization limitation which leads to inexact simulation of extreme flow events. One possible reason for the discrepancies is the selection of the training data sets. Some solutions proposed for this problem from the earlier studies (Minns & Hall 1996; Imrie *et al.* 2000) are careful scaling of calibration data and include more high flow patterns in the training set. Another possible reason for the discrepancies are the errors resulting from flow measurements.
2. Representative details of the hydrographs are presented in Figure 6, in which the capability of predicating peak events by the three models could be obviously viewed. The SVR estimates of the peak and low values are closer to the corresponding observed values than those of the ANN and the ELM model. The ANN model is found to be over-predicting the peak flows of extreme, while the EML model seriously underestimates a number of moderate and high magnitudes of the flows.
3. From the overall exercises, the possibility of using the ANN, SVR, ELM data-driven components for river flow modeling was demonstrated. The fact that the three data-driven components ran independently and sufficiently good forecasting results were made prove that the system can be applied to real world scenarios.

CONCLUSIONS

In this paper we presented a flexible system for building hydrological forecasting applications with data-driven models. The development of the system has been motivated by the following considerations: low modeling efficiencies and complex programming that block the current practice in hydrology forecasting applications; the lack of approaches that facilitate reuse of existing codes and models; the diversity of observations data sources and the heterogeneity problems that make the data not easy to use. To resolve these issues, the following work was carried out. The problem of accessibility of modeling for non-programmers was addressed by developing the visual modeling system. To facilitate reuse of the existing model, a set of standard interfaces for data-driven modeling were defined.

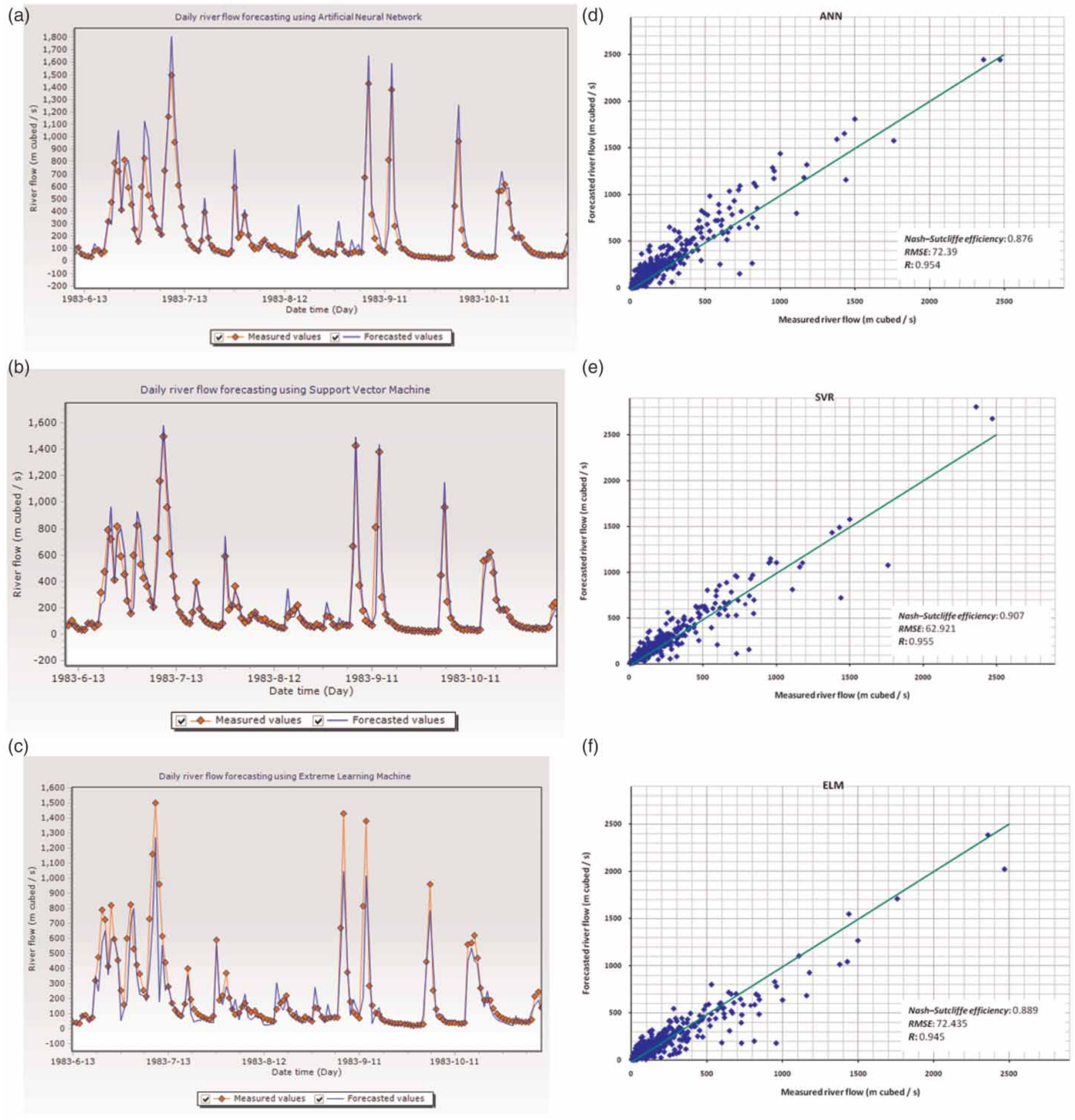

Figure 6 | Representative detail of hydrographs and scatter plots of the 1-day ahead river flow forecasting results produced by the ANN model (a and d), the SVR model (b and e), and the ELM model (c and f).

Components that support the interfaces become acceptable by the system. Existing models written in different programming languages could be migrated to the system through a wrapping pattern. To enhance organization and analysis of point observations data, the ODM data model was adopted, upon which a variety of tools and utilities were developed to assist the user in managing and analyzing data and pre-and-post processing data in modeling procedures.

The presented system is still under continuous development. In future work we will address the following issues. (1) More data-driven methods will be employed in the near future, such as genetic programming (GP) and evolutionary regression, regression trees and model trees, etc. (2) Conceptual hydrological models such as Sacramento Soil Moisture Accounting (SAC-SMA) are considered to be included in the system in order to give the users more choices and to facilitate comparison of performance of different types of models. (3) The accuracy of a data-driven or conceptual model is largely dependent on selection of the model parameters. However, structured methods for selecting parameters are not provided by the system. Recently, various methods have been proposed to identify the parameters, such as the simulated annealing algorithms, the shuffled complex evolution algorithm (SCE-UA) and the non-dominated sorting genetic algorithm (NSGA-II). These methods will be employed to assist with identifying the optimal model parameters. (4) The linkage mechanism of the current system is far from perfect. Sometimes a forecasting model requires an input value that is generated by other models. This requires an enhanced linkage mechanism that allows for models from different background and approaches working in concert.

ACKNOWLEDGMENTS

This work was supported by a grant from the Special Research Foundation for the Public Welfare Industry of the Ministry of Water Resources (Project No. 201001080) and a grant from the Fundamental Research Funds for the Central Universities (Program No. 2012TS017).

REFERENCES

Ahuja, L. R., Ascough Ii, J. C. & David, O. 2005 Developing natural resource models using the object modeling system: feasibility and challenges. *Adv. Geosci.* **4**, 29–36.

Banerjee, P., Singh, V. S., Chatttopadhyay, K., Chandra, P. C. & Singh, B. 2011 Artificial neural network model as a potential alternative for groundwater salinity forecasting. *J. Hydrol.* **398** (3–4), 212–220.

Basheer, I. A. & Hajmeer, M. 2000 Artificial neural networks: fundamentals, computing, design, and application. *J. Microbiol. Meth.* **43** (1), 3–31.

Bray, M. & Han, D. 2004 Identification of support vector machines for runoff modelling. *J. Hydroinform.* **6** (4), 265–280.

Chau, K. W., Wu, C. L. & Fan, C. 2010 Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques. *J. Hydrol.* **389** (1–2), 146–167.

Cherkassky, V. & Ma, Y. Q. 2004 Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Netw.* **17** (1), 113–126.

Choi, T. M., Sun, Z. L., Au, K. F. & Yu, Y. 2008 Sales forecasting using extreme learning machine with applications in fashion retailing. *Decis. Support Syst.* **46** (1), 411–419.

David, O., Markstrom, S. L., Rojas, K. W., Ahuja, L. R. & Schneider, I. W. 2002 The object modeling system. In: *Agricultural System Models in Field Research and Technology Transfer* (L. Ahuja, L. Ma & T. Howell, eds). CRC Press, Boca Raton, FL, pp. 317–330.

Dibike, Y. B. & Solomatine, D. P. 2001 River flow forecasting using artificial neural networks. *Phys. Chem. Earth B–Hydrol. Oceans Atmos.* **26** (1), 1–7.

Dibike, Y. B., Velickov, S., Solomatine, D. & Abbott, M. B. 2001 Model induction with support vector machines: introduction and applications. *J. Comput. Civil Eng.* **15** (3), 208–216.

Environmental Data Standards Council 2006 *Environmental Sampling, Analysis, and Results Data Standards: Overview of Component Data Standards*, Standard No.: EX000001.1. The Environmental Data Standards Council (EDSC), US EPA, Washington, DC. (Available at www.exchangenetwork.net/standards/ESAR_Overview_01_06_2006_Final.pdf.)

Fausett, L. 1994 *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Gopakumar, R., Takara, K. & James, E. J. 2007 Hydrologic data exploration and river flow forecasting of a humid tropical river basin using artificial neural networks. *Water Resour. Manage.* **21** (11), 1915–1940.

Govindaraju, R. S. & Artific, A. T. C. A. 2000 Artificial neural networks in hydrology. I: Preliminary concepts. *J. Hydrol. Eng.* **5** (2), 115–123.

Holzworth, D. P., Huth, N. I. & de Voil, P.G. 2010 Simplifying environmental model reuse. *Environ. Model. Softw.* **25** (2), 269–275.

Hong, W. C., Dong, Y. C., Chen, L. Y. & Wei, S. Y. 2011 SVR with hybrid chaotic genetic algorithms for tourism demand forecasting. *Appl. Soft Comput.* **11** (2), 1881–1890.

Horsburgh, J. S., Tarboton, D. G., Maidment, D. R. & Zaslavsky, I. 2008 A relational model for environmental and water resources data. *Water Resour. Res.* **44** (5), 1–12.

Horsburgh, J. S., Tarboton, D. G., Piasecki, M., Maidment, D. R., Zaslavsky, I., Valentine, D. & Whitenack, T. 2009 An integrated system for publishing environmental observations data. *Environ. Model. Softw.* **24** (8), 879–888.

Huang, G.-B. & Chen, L. 2008 Enhanced random search based incremental extreme learning machine. *Neurocomputing* **71** (16–18), 3460–3468.

Huang, G. B., Zhu, Q. Y. & Siew, C. K. 2006 Extreme learning machine: theory and applications. *Neurocomputing* **70** (1–3), 489–501.

Imrie, C. E., Durucan, S. & Korre, A. 2000 River flow prediction using artificial neural networks: generalisation beyond the calibration range. *J. Hydrol.* **233** (1–4), 138–153.

Jeton, A. E. 2001 Streamflow forecasting using the modular modeling system and an object-user interface. In: *Proceedings of the Western Snow Conference, 69th Annual Meeting*. National Resources Conservation Service, USA, pp. 85–91.

Lan, Y., Soh, Y. C. & Huang, G.-B. 2010 Two-stage extreme learning machine for regression. *Neurocomputing* **73** (16–18), 3028–3038.

Leavesley, G. H., Markstrom, S. L. & Viger, R. J. 2000 The Modular Modeling System (MMS) – The physical process modeling component of a database-centered decision support system for water and ecosystem management. Second Symposium on Environmental Applications. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 92–97.

Liong, S. Y. & Sivapragasam, C. 2002 Flood stage forecasting with support vector machines. *J. Am. Water Resour. Assoc.* **38** (1) 173–186.

Lorrai, M. & Sechi, G. M. 1995 Neural nets for modelling rainfall-runoff transformations. *Water Resour. Manage.* **9** (4), 299–313.

Lowy, J. 2005 *Programming .NET Components*, 2nd edition. O'Reilly Media, Inc., USA.

Martínez-Martínez, J. M., Escandell-Montero, P., Soria-Olivas, E., Martín-Guerrero, J. D., Magdalena-Benedito, R. & Gómez-Sanchis, J. 2011 Regularized extreme learning machine for regression problems. *Neurocomputing* **74** (17), 3716–3721.

Minns, A. W. & Hall, M. J. 1996 Artificial neural networks as rainfall-runoff models. *J. Hydrol. Sci.* **41** (3), 399–417.

Moore, R. V. & Tindall, C. I. 2005 An overview of the open modelling interface and environment (the OpenMI). *Environ. Sci. Pol.* **8** (3), 279–286.

Narendra, K. S. & Parthasarathy, K. 1990 Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1** (1), 4–27.

Pai, P. F., Lin, K. P., Lin, C. S. & Chang, P. T. 2010 Time series forecasting by a seasonal support vector regression model. *Expert Syst. Appl.* **37** (6), 4261–4265.

Rizzoli, A. E., Davis, J. R. & Abel, D. J. 1998 Model and data integration and re-use in environmental decision support systems. *Decis. Support Syst.* **24** (2), 127–144.

Sivakumar, B., Jayawardena, A. W. & Fernando, T. M. K. G. 2002 River flow forecasting: use of phase-space reconstruction and artificial neural networks approaches. *J. Hydrol.* **265** (1–4), 225–245.

Smola, A. J. & Schölkopf, B. 2004 A tutorial on support vector regression. *Statist. Comput.* **14** (3), 199–222.

Solomatine, D. P. & Ostfeld, A. 2008 Data-driven modelling: some past experiences and new approaches. *J. Hydroinform.* **10** (1), 3–22.

Tarboton, D. G., Horsburgh, J. S. & Maidment, D. R. 2008 CUAHSI Community Observations Data Model (ODM) Version 1.1 Design Specifications. (Available at http://his.cuahsi.org/documents/ODM1.1DesignSpecifications.pdf.)

Vapnik, V. N. 1998 *Statistical Learning Theory*. Wiley, New York.

Vapnik, V. N. 1999 An overview of statistical learning theory. *IEEE Trans. Neural Netw.* **10** (5), 988–999.

Yu, P. S., Chen, S. T. & Chang, I. F. 2006 Support vector regression for real-time flood stage forecasting. *J. Hydrol.* **328** (3–4), 704–716.

First received 23 October 2011; accepted in revised form 7 May 2012. Available online 20 July 2012