AN ADAPTIVE SYSTEM FOR WEB SERVICES THROUGHPUT PREDICTION

Răzvan Daniel Albu

University of Oradea, Department of Electronics, Faculty of Electrical Engineering and Information Technology, ralbu@uoradea.ro

ABSTRACT: Adaptive systems in engineering are an interdisciplinary field that deals with distributed, nonlinear systems. They harmonize present engineering design principles and are suitable to interface directly with the real world with little or no pre-processing. When applied correctly, such systems can significantly do better than more traditional techniques. In this article I studied an adaptive model for web services throughput prediction using data of 5825 real-world Web Services. Detailed experimental results are also presented in this paper to validate the performance of the model. Using Synapse adaptive model simulator I obtained a prediction error of + 9.05139 with 99% confidence.

KEY WORDS: web services, adaptive systems, recurrent neural networks, throughput prediction.

1. INTRODUCTION

An adaptive system is a system that is able to adjust its behaviour according to changes in its environment or in parts of the system itself. Adaptive systems are self-modifying mathematical models that act as intelligent agents that can find complex patterns in data are able to make precise predictions and are capable to learn from mistakes. In an era when the amount of data is growing exponentially, adaptive systems are a revolution in the way we handle and appreciate information. With adaptive systems we can do better, faster and more sustainable than what could have been imagined only a few years ago.

A web service is nothing with which an end user would directly interact, but it is a platform for developing interoperable distributed applications that allows a programmer to interact with other information providers, without worrying about what they are running either at the backend or even their front-end. Web services have been promising in modern times and are by now one of the most popular method for constructing resourceful distributed systems. The performance of the service oriented systems is highly relying on the performance of the employed Internet Web services. Because of the popularity of Web services on the Internet, studying the quality and reliability of Web services is becoming more and more essential. Web services can be discovered from three main sources:

- UDDI (Universal Description, Discovery and Integration, which is an XML based registry enabling companies to publish and discover Web services on the Internet),
- Web service portals (e.g., xmethods.net, webservicex.net, webservicelist.com, etc.);
- Web service searching engines (e.g., seekda.com, esynaps.com, etc.).

SOAP is a XML-based (Extensible Markup Language), lightweight protocol for the exchange of information in a decentralized, distributed environment that consists of three parts [1]:

- An envelope that defines a framework for describing what is in a message and how to process it;
- A set of encoding rules for expressing instances of application-defined data types;
- A convention for representing remote.

2. RELATED WORKS

Analyzing the service computing literature [5], a number of adaptive models driven approaches have been elaborated for service selection [9], optimal service composition [6], fault tolerant Web services [13] and so on. However, there is still a need of real-world Web service datasets for validating new techniques and models.

Traditional schools such as: CSR (Centre for Software Reliability) City University London, University of Calgary in Canada, University of Lübeck in Germany or University of Birmingham dedicated important resources to this subject. In [14] were provided some reusable research datasets and built several large-scale evaluations on real-world Web services for promoting the research [11]. Browsing the literature we can find many neural network based software reliability prediction models and their prediction capability is confirmed better than some statistical models.

Based on a study of 50 scientific papers published in high quality journals and 35's proceeding of international conferences, I noticed that works on software reliability prediction models, from 1991 to 2011, has undergone a spectacular evolution that is shown in Fig. 1.

Karunanithi was among the first who presented a model of software reliability prediction based on neural networks [7]. They used the execution time as input of neural network. They have also tested their model in different network architectures such as feed-forward or recurrent neural networks.
Sitte presented a method based on neural networks for prediction and compared it with the parametric recalibration models using several significant metrics on the same data sets [8]. He concluded that the neural network approach provides better predictions. Cai proposed a prediction method based on NNs using the Backpropagation algorithm for training [4]. They used data from the last 50 errors as inputs to predict failure the next time out. They concluded that the effectiveness of the method depends strongly on the nature of the dataset used. Su and his team proposed the DWCM (Dynamic Weighted Combination Model) based on neural networks to predict software reliability [10]. They used different activation functions in hidden layers depending on SRGM (Software Reliability Growth Models) and applied it on two data sets by comparing the results with statistical models. Experimental results show that DWCM gives better results than traditional models. In [2] the authors have derived SRGM models based on NHPP heterogeneous processes (Non-homogeneous Poison Processes) using a unified theory by introducing the concept of "multiple change points" in software reliability modelling. They estimated the parameters of the proposed models using three datasets of software failures and the results were compared with those of existing SRGM models. Their model predicted cumulative number of failures at different stages of development and operation of software.

3. ADAPTIVE MODEL

In fig. 2 is illustrated the model I developed and used for throughput prediction. Moving from left to right it is compound of: input data source, a gamma memory block, a function layer block, a weight layer block, another function layer block, a delta terminator and the output data source. As we can see, there is also a merger that acts as a multiplexer. It adds the features from the input signals collected from the output data source and from the last function layer, to the common output signal which will be graph on the Value/Simple plot block. The gamma memory is used in dynamic systems to remember past signals. It enables the usage of past information to predict current and future states. Its primary parameters are shown in table 1.

The number of outputs equals the number of input features times the number of taps. Taps is the number of tap delays; $\mu$ is the gamma scaling factor, when is set to 0 the gamma memory is a simple tapped delay line. One must keep in mind that by adapting the $\mu$ parameter, we are changing the poles of the system, which can lead to unstable solutions. Gamma is stable for $0 < \mu < 2$.

In [12] the authors collected a 339 * 5825 user-item matrix with web-services throughputs. In this experiment from this huge matrix I used just the first three columns so the input data source block contains the first two columns (718 samples). The function layer is a block that applies a bias and a function to its input. The bias is controlled through a set of weights which can be adapted. The first function layer has 8 inputs and 8 outputs. Its activation function is tanh (Hyperbolic tangent). The general operation of the function layer is described in (1):

$$Y = f(X + b)$$

(1)

Where $Y$ is the output, $X$ the input, $b$ the bias and $f$ the activation function

The weight layer block is a fundamental form of long-term memory for an adaptive system. Each input feature is connected to each output feature and scaled by a weight. The weight layer is used as a container of parameters that can be adapted through training. These weights scale the inputs and thus perform a filtering operation. The general operation of the weight layer is described in (2):

$$Y = X \ast W$$

(2)

The weight layer supports a simple heuristic regularization method called "Brain damage". The principle is that the weights are randomly perturbed in small amounts in order to force a stable solution. If a small disturbance of the weights causes a big change then the solution was probably stuck in a local minimum and would benefit being pushed out of it. The noise level is the boundary for initial weight randomization. Its primary parameters are shown in table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>8</td>
</tr>
<tr>
<td>Outputs</td>
<td>1</td>
</tr>
<tr>
<td>Brain damage</td>
<td>false</td>
</tr>
<tr>
<td>Noise level</td>
<td>0.3535539059327373</td>
</tr>
</tbody>
</table>

The last function layer has one input and one output and the same tanh activation function. The output data source contains the third column that contains desired throughput values. The value/sample plot is used to plot signals or part of signals in the order of the samples. The “Lazy Epoch” buffer mode measures the buffer length in epochs. When the buffer is full it is emptied and the contents written to the format. Its primary parameters are shown in table 3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>2</td>
</tr>
<tr>
<td>Buffer Mode</td>
<td>Lazy Epoch</td>
</tr>
<tr>
<td>Set</td>
<td>Training</td>
</tr>
<tr>
<td>Flood Limit</td>
<td>1000</td>
</tr>
</tbody>
</table>

The set parameter selects what data set to display: training or validation. The flood limit is the maximum number of samples to display at once. This adaptive model was trained 100 epochs with a batch length of 359. The batch length expresses how many samples to take in to account for each update of the system parameters. Theory says that a larger number of
samples improves reliability of statistics used in calculations and often gives smoother convergence. The batch length should preferably be chosen so that the full length is a multiple of the batch length. Failing to have a batch length that fits exactly can result in samples being cropped. While this is of little consequence for static systems, it may result in disruptions between the training and validation sets when dealing with dynamic systems.

Figure 2. Design of the adaptive model

4. CASE STUDY

The development cycle in Synapse is based on the entrenched development process model for adaptive systems. The difference is that traditionally diverse tools were used for the individual steps while Synapse offers a exclusively integrated environment that sustains the whole cycle within one reliable application. This and progress in computer software and hardware allows Synapse to make the growth cycle more efficient, interactive, easier to use, and significantly more powerful.

To train the adaptive model I use the Web service QoS dataset built by [3] They collected a 339 * 5825 user-item matrix for throughput, where an entry Ra,i in the matrix is the throughput of Web service i observed by the service user a:

Table 4. Statistics of the dataset.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of Web Service Invocations</td>
<td>1,542,884</td>
</tr>
<tr>
<td>Num. of Service Users</td>
<td>150</td>
</tr>
<tr>
<td>Num. of Web Services</td>
<td>100</td>
</tr>
<tr>
<td>Num. of User Countries</td>
<td>24</td>
</tr>
<tr>
<td>Num. of Web Service Countries</td>
<td>22</td>
</tr>
<tr>
<td>Range of Failure Probability</td>
<td>0-100%</td>
</tr>
</tbody>
</table>

Table 4 shows some interesting statistics about the dataset to prove its relevance. In the experiment, the system is trained using 85% of the data and the remaining 15% data are used for testing. The data set contains also two matrices stored in the following text files:

Table 5. Structure of userlist.txt and wslist.txt.

<table>
<thead>
<tr>
<th>userlist.txt 339 service users</th>
<th>19KB</th>
<th>User ID</th>
<th>IP address of user</th>
<th>country</th>
<th>longitude</th>
<th>Latitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>wslist.txt 5825 Web services</td>
<td>505KB</td>
<td>WS ID</td>
<td>WSDL address</td>
<td>provider name</td>
<td>country name</td>
<td></td>
</tr>
</tbody>
</table>

For the 359X5825 matrix with throughput values, mean and standard deviation is 102.86 kbps and 531.85 kbps, respectively. Based on three-sigma rule this is a Gaussian distribution. In this experiment I used the first three columns for training. Some interesting statistics about them are presented in table 6 and their graphical representation in fig 3:

Table 6. Statistics about the training data set.

<table>
<thead>
<tr>
<th>Column 0</th>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max throughput</td>
<td>4.629</td>
<td>51.282</td>
</tr>
<tr>
<td>Mean</td>
<td>1.509135</td>
<td>14.10798</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.9159551</td>
<td>4.89436</td>
</tr>
</tbody>
</table>
Batch Processors are a general class of automation methods. They can be used for instance to adapt systems remotely over network or to optimize system structures. To train the input data I used a "Genetic Optimizer" batch Processor. Its settings are shown in fig. 4.

The optimizers try to find optimal values for various parameters of the system, but not the weights of the system. To save time it is common to train less during optimization than you would otherwise. For instance the system in the example picture above was trained for 100 epochs after optimization. In the Synapse error analyzer in Confidence view, we can see that we managed to achieve our requirements, the output error is $\pm 9.05139$ with 99% confidence.

5. CONCLUSIONS AND FUTURE RESEARCH

The model described in this paper can predict the throughput of a web service very accurately. The situation of arguing that a new model have higher predictive accuracy than do others, share the issue that performances are being calculated and compared in different test sets. The accuracy will differ across test sets.

This makes it impossible to define the accuracy of a prediction model independent of the test set to which it is applied. As a result, to make a correct evaluation the new prediction models must be compared to existing models on the same data sets. As future research I plan to investigate the prediction accuracy of this adaptive model with different data sets and various inputs and also compare it to other models on the same data set.

6. ACKNOWLEDGEMENTS

This work was partially supported by the strategic grant POSDRU/CPP107/DM11.5/S/80272, Project ID80272 (2010), co-financed by the European Social Fund-Investing in People, within the Sectorial Operational Program Human Resources Development 2007-2013.
7. REFERENCES

1. (W3C recommendations 2007) http://www.w3.org/TR/.