Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

FEATURE-BASED GENERATORS FOR TIME SERIES DATA

Jorge R. Ramos
Vernon Rego

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, U.S.A.

ABSTRACT

A variety of interesting domains, such as financial markets, weather systems, herding phenomena, etc., are characterized by highly complex time series datasets which defy simple description and prediction. The generation of input data for simulators operating in these domains is challenging because process description usually involves high-dimensional joint distributions that are either too complex or simply unavailable. In such applications, a standard approach is to drive simulators with (historical) trace-data, along with facilities for real-time interaction and synchronization. But, limited input data, or conversely, abundant but low-fidelity random data, limits the usefulness and quality of the results. With a view to generating high-fidelity, random input for such applications, we propose a methodology which uses the original data, as a template, to generate candidate datasets, to finally accept only those datasets which resemble the template, based upon parameterized features. We demonstrate the methodology with some early experimental results.

1 INTRODUCTION

Simulators typically rely on well-known statistical distributions (Law and Kelton 2000) to generate input data. These distributions have theoretical formulations which allow analysts to describe their behaviour in precise, mathematical and probabilistic terms. Their main usefulness in simulation stems from the fact that they can often be used to describe real-world observations in a fairly accurate, succint and simple way. When theoretical distributions are not appropriate, real-world data can be used to construct empirical input distributions. A simulator may then recreate variations on the behavior of a real-world system by varying input parameters. Independence is often a feature of the data-generation scheme for simplicity or by necessity.

Time series data occurs in a variety of interesting domains but, because of complex dependencies in time, is not easily described by standard distributions. When such data is required in abundance, with features of reproducability and parametric control, random data-generation can be challenging. Examples of systems which require such data include financial price series for stocks, bond and options, commodity prices, weather systems, irregular but cyclic data for productivity measures, economic cycles, etc. In the case of stock prices, for example, one approach is to use real-time or historical market data combined with user-generated decisions in the simulator (Kearns and Ortiz 2003). Here, to preserve data integrity, the market is polled at certain instants in time and, with the aid of a set of rules, instantaneous market data is synchronized with simulator output. While this tack can produce interesting insights into the nature of the observed system – in this case, a market on stocks – it is far from ideal because it provides only a single realization of a price trajectory and lacks reproducability. As a result, aggregate statistical analysis cannot be done on the output data; further, conclusions drawn from use of a single trajectory are subject to questions of data-snooping (Sullivan, Timmermann, and White 1998). In studying the efficacy of trading algorithms, for example, a single dataset is hardly adequate. Conversely, the use of purely random input data has such a low-fidelity, that it will offer poor results.

The time series data generation problem arose in an application involving the information content of manipulated data (Ramos and Rego 2005) in the financial markets. It was necessary to process randomly generated clones of a real-world dataset (i.e., historical stock/options data) for studies in classification. While it has been suggested that stock data has a random walk property (Malkiel 2004), in the sense that the average person may be hard-pressed to distinguish between a real stock price series and one generated using, say, the method of independent increments with Gaussian variates, an experienced trader would have little difficulty telling the data apart, even without additional clues (name of stock, size of float, etc). This is because a stock's price series tends to exhibit characteristic patterns or "structure" that is not easily captured by naive data-generation schemes. This

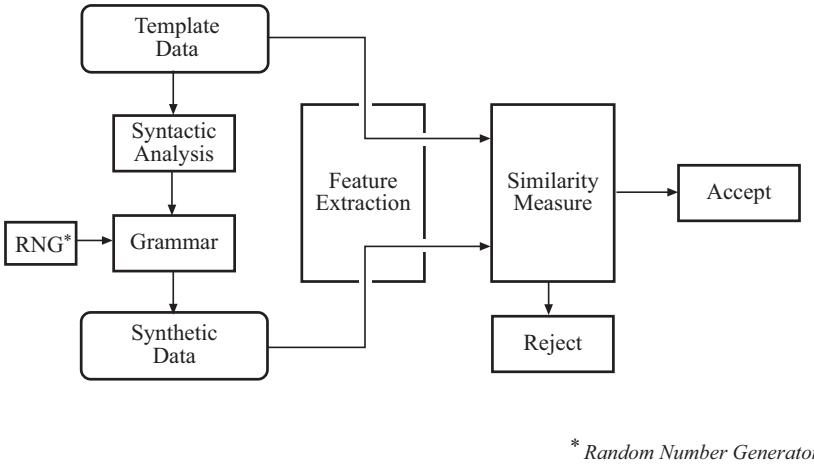

Figure 1: Feature Based Generation of Time Series

includes high-frequency oscillations due to local overbought and oversold conditions, spikes, and also broader chart patterns (cup and handle, saucer, head and shoulders and double-bottom formations, etc).

Time series data generated by simple methods which ignore structure exhibit characteristics that are inherently different in appearance from data generated by methods which attempt to account for structure.

We present a methodology to generate random instances of time series which resemble a given template with specifiable measures of similarity on critical features. That is, the generated data is random but exhibits similarity to given historical or synthetic data; the template is used as a basis for generation, so that the resulting datasets are similar. There are two ways to proceed: (a) generate random data with select template features, and (b) generate random datasets with some template property but accept only those datasets which exhibit features similar to template features. In this paper we focus on method (b), because the emphasis of our original application was classification; by using pattern recognition and data mining algorithms, it is possible to classify series data based on selectable and relevant features (Last, Klein, and Kandel 2001; Gavrilov et al. 2000). We present method (a) in another paper.

An overview of the generation process is shown in Figure 1. The data trajectory in the given template is quantized, and then syntactically analyzed, to generate a probabilistic grammar that can generate sentence data that resembles the template. Generating a random data trajectory is equivalent to generating a random sentence using the grammar. The template and synthetically generated data are subjected to characterization by a feature extraction module which transforms trajectories into characteristic vectors. A similarity measure is used to classify/select synthetically generated trajectories that are acceptable, by exhibiting patterns resembling the template. By using feature extraction techniques and weighed similarity measures, we are able to quantify the likenes of synthetically generated datasets to generating template. A set of parameters enable an analyst to vary the desired degree of similitarity between template and synthetic dataset. The acceptance procedure is parameterized in that it utilizes a number of relevant features, with each requiring a specifiable threshold that functions as a similarity measure.

The remainder of the paper is organized as follows. In Section 2, we explain how probabilistic context free grammars can be used to generate data. In Section 3, we examine some important feature extraction algorithms for time series. Section 4 shows how to use the features to determine similarity between a template and its synthetic trajectories. In Section 5, we show some experimental results using actual time series data. A brief conclusion is given in Section 6.

2 DEPENDENCY STRUCTURE

A number of distinct methods may be used to capture the structure of dependency in the original data for the purpose of sampling trajectories. We chose to use techniques of syntactic pattern recognition (Fu 1982) and formal language theory – grammars and production rules – to describe dependencies, because they offer use of convenient Markov properties, and because grammar-based generation was crucial in our original application (Ramos and Rego 2005). Syntactic pattern recognition is well-suited to classification studies using time-based data, where a relationship between symbols is an essential feature requiring description. This is an off-shoot of a methodology for data-recognition; we see it as a middle-ground approach, lying between use of theoretical distributions and accurate summaries of dependence

for sequences of random variables, and purely unstructured real-time observations or histograms.

A grammar formally describes a language using a relatively small set of rules called productions; it is defined as a quadruple $G =< \Sigma_T, \Sigma_N, P, S >$ (Friedman and Kandel 1999), where Σ_T is a set of terminal symbols, Σ_N is a set of non-terminal symbols, P is a set of production rules and S is the starting symbol (or root). Σ_T and Σ_N are subsets of an alphabet Σ, which is a finite set of symbols $x_1, x_2, ..., x_n$.

A production rule for a context-free grammar has the form:

$$A_1 \rightarrow \beta_2,$$

where the symbol A_1 undergoes substitution by string β_2. cfg's require a non-terminal (i.e. $A_1 \in \Sigma_T$) on the left side of the production rule, with no restrictions on the right side. That is, β_1 can be a combination of terminals and non-terminals. cfgs have played a significant role in the development of computer sciences, where they have been used in computational theory for abstract machines (Brookshear 1989) and, more practically, for the specification of computer languages and the building of compilers (Aho, Sethi, and Ullman 1986).

Stochastic (or probabilistic) grammars (Friedman and Kandel 1999) allow for some ambiguity in the patterns, by associating a probability Q with a set of productions. A stochastic grammar is defined by a quintuple $G =< \Sigma_T, \Sigma_N, P, Q, S >$.

Probabilistic context free grammars (PCFG) are stochastic extensions of context-free grammars and are used to create models of semistructured and ambiguous data by use of machine learning algorithms (Pearl 1988, Mitchell 1997). PCFGs (Stolcke and Omohundro 1994, Pynadath and Wellman 1998) have been used for pattern recognition in many areas, such as speech recognition (Jurafsky et al. 1995), music (Steedman 1984) and RNA modelling (Sakakibara et al. 1994). A PCFG is formed by generating hierarchical production rules in the form of trees, with associated probabilities, based on a set of learning data.

We use the following algorithms to process a template and generate synthetic time series data:

1. **Analysis for PCFG generation**: Consecutive data increments in the original data template are Normally distributed and offer a convenient mechanism for PCFG symbol generation.
   We begin by computing differences between consecutive data points, while storing the absolute value of the starting point p_0. A quantization constant q is chosen to map numerical values into symbols, by dividing the range of numbers into s equally sized buckets of size q; clearly, this yields a total of s symbols for the sequence. Given a set of symbols, we proceed to the next step, which is to capture dependency. While this can be done in a number of ways, we use syntactic analysis to construct conditional probability tables for $m-$th order Markov transitions. With $m = 1$, these tables offer the empirical probability of symbol A_i being followed by symbol A_j, $1 \leq i, j \leq s$.
   A PCFG is obtained, with production rules similar to these examples:

   $A_1 \rightarrow A_2 \mid A_3 \mid A_6$ (0.25 | 0.45 | 0.30)
   $A_2 \rightarrow A_5 \mid A_1 \mid A_7 \mid A_9$ (0.70 | 0.10 | 0.15 | 0.05),

   which means that A_1 is followed by A_2 with probability 0.25, by A_3 with probability 0.45, etc.
2. **Synthetic data generation**: On obtaining the PCFG for a given template, we arrive at a tree whose paths represent transitions between consecutive symbols, and path probabilities are transition probabilities. We may then then choose a suitable starting symbol for the new sequence (say, starting symbol p_0 from the template). The new sequence is generated by following the production rules, always choosing the next symbol using a uniform variate and the given conditional probabilities. Once a sentence (symbol sequence) is ready, we must invert symbols into numerical (time series) values. Some information is lost during quantization, and a uniform variate is used to generate a value within a symbol's bucket range. Recalling that the grammar is based on increments, the conversion from symbols to numerical data (starting from p_0) requires consecutive additions for generation of the synthetic series.

3 FEATURE EXTRACTION

Computational mining of time series data has grown in importance in recent years. Algorithms have been developed to clean, compress, index and detect certain patterns in particular time series. In the sequel we exploit a few of these techniques to help quantify useful template data features that we would like preserved in the synthetsized data.

Time series data is often noisy and short-term variations can obscure long-term trends. A first step usually entails cleaning of data to to eliminate as much noise as possible while retaining important features; this is done with moving averages, filters, and transformations/selection of certain points. The transformed data is then used to obtain features of interest. Among methods available in the literature to characterize time series, we have found the following to be useful in our context:

1. **Slope, Length, & Signal to Noise Ratio**: A key feature of a time series is its overall trend, character-

ized by its slope. We use the algorithm presented in (Last, Klein, and Kandel 2001) to clean the data using a finite impulse response (FIR) filter, interpolate to eliminate local extrema lying within time-threshold d, and finally merge consecutive data segments with slopes that differ by less than some threshold ϵ. Other measures given by this algorithm include segment lengths (in time units) and signal to noise ratio, which is the ratio between the original data values and newly generated segments. User specified parameters include d, ϵ, and bandwidth B and number of coefficients N of the filter. We refer to the features derived here as SLOPE, LENGTH and SNR.

2. **Extremal points & indexing values**: We use the algorithm in (Fink and Pratt 2003) to locate major extrema based on a parameter R. The consecutive segments ("legs") formed by connecting extrema are used to calculate certain indexing values. If VL and VR represent left and right end-points of a leg, then RATIO=VR/VL. This algorithm can also provide leg-length in time units, though we we only use VL and RATIO (since the VR of one segment is the VL of the following segment). We use the quantity LENGTH given by the algorithm quoted in the prior step.
3. **Distribution characteristics of increments**: Consecutive incremements in the time series data is approximately Normal. Estimates of skew, kurtosis and mean of the distribution can be used to quantify the similitarity between template data and synthetic data; standard deviation is not useful for making distinctions because it has experimentally been shown to offer like values for distinct series. We refer to the features computed here as SKEWNESS_Δ, KURTOSIS_Δ and MEAN_Δ.

The algorithms generate values that help characterize a given time series. By collecting these values in a feature vector, we effectively obtain a signature for each time series. Some algorithms generate multiple m values for one feature (i.e., SLOPE, RATIO, etc.) in one data trajectory, and $n \leq m$ values in a second data trajectory. In this case we use the first n values in the feature vector. We use the term FEATURE(i) to denote the i-th feature.

4 SIMILARITY MEASURES

To compare the feature vectors we expand on similarity measures developed in (Fink and Pratt 2003), based on a zero-to-one scale, with zero implying no likeness and one implying complete likeness. Having a uniform scale, as opposed to distance-based methods, makes it easier to establish criteria for accepting or rejecting synthetic data — a threshold selected in the 0-1 range will suffice.

Similarity between two numbers a and b is defined as:

$$\mathcal{S}(a,b) = 1 - \frac{|a-b|}{|a|+|b|}.$$

The *peak similarity* between two points, a and b is:

$$\mathcal{S}^*(a,b) = 1 - \frac{|a-b|}{2 \cdot max(|a|,|b|}.$$

The *mean similarity* and *root mean square similarity* between two series $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are given by:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{S}(a_i, b_i), \text{ and,}$$

$$\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{S}(a_i, b_i)^2},$$

with similar definitions for *peak similarity* and *root peak square similarity* between series.

While the similarity measures are usually applied to a uniform vector space, in our case, however, dissimilar features are combined in a single vector. Furthermore, some features are more relevant than others and this asymmetry should be observed in the final result. To effect this, we introduce weighted similarity measures:

$$\mathcal{S}_i(a,b) = 1 - w_i \cdot \frac{|a-b|}{|a|+|b|},$$

$$\frac{1}{W} \cdot \sum_{i=1}^{n} \mathcal{S}_i(a_i, b_i),$$

where

$$W = \sum_{i=1}^{n} w_i.$$

The definitions for all other corresponding *weighted* similarity measures are straightforward. Clearly, if all weights w_i are equal, we obtain the original measures.

5 EXPERIMENTAL RESULTS

We present some typical results that show the practical application of our methodology, and address issues an analyst must consider in choosing parameters in designing a classifier. Our template contains a single year of daily closing prices of Microsoft's stock (MSFT), starting in February 2000, as shown in Figure 2. The chosen region shows price fluctuations overlaying a clear trend, which makes it an interesting dataset for experiments.

5.1 Experimental Setup

A PCFG tree was generated using the above template with s=15 symbols. If s is too small the generated data will show bigger fluctuations than the template, while too big a number will reduce transition variability, and thus reduce randomness. Independent price trajectories were generated with the help of the PCFG tree and independent random

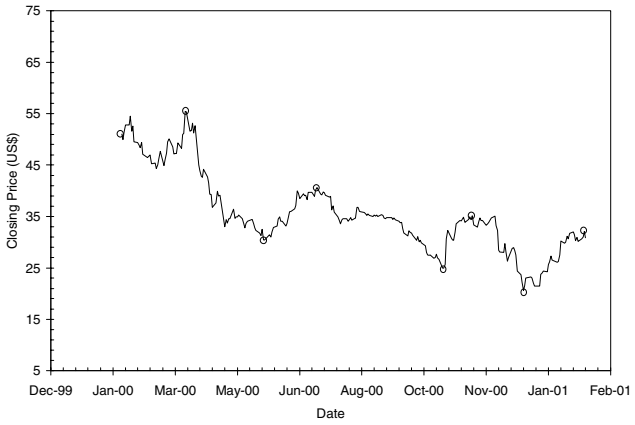

Figure 2: Template, MSFT Stock with Extremal Points

number seeds. We show three examples in Figures 3, 4 and 5. Each one of these trajectories represented a candidate dataset, to be accepted or rejected based on parametrized features.

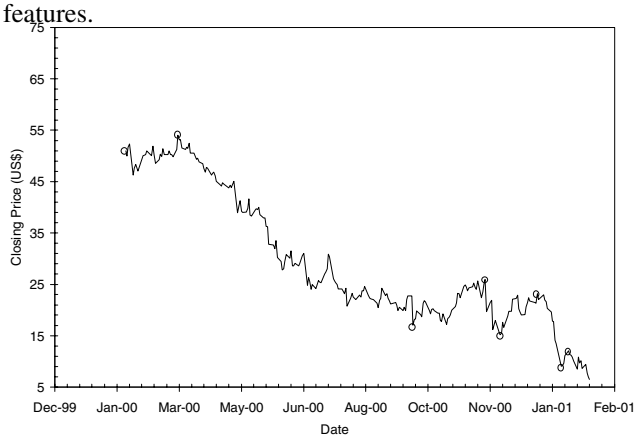

Figure 3: Synthetic Trajectory 1 with Extremal Points

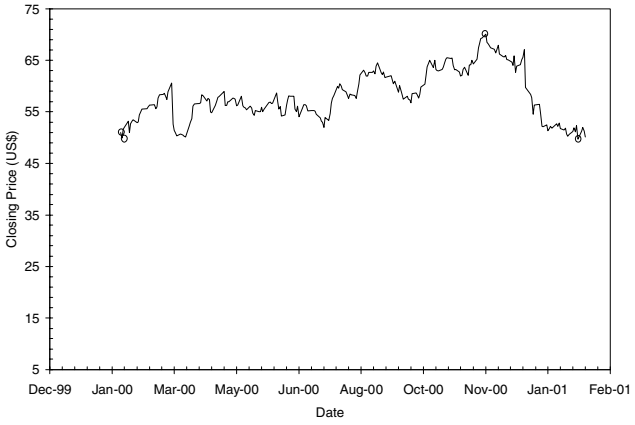

Figure 4: Synthetic Trajectory 2 with Extremal Points

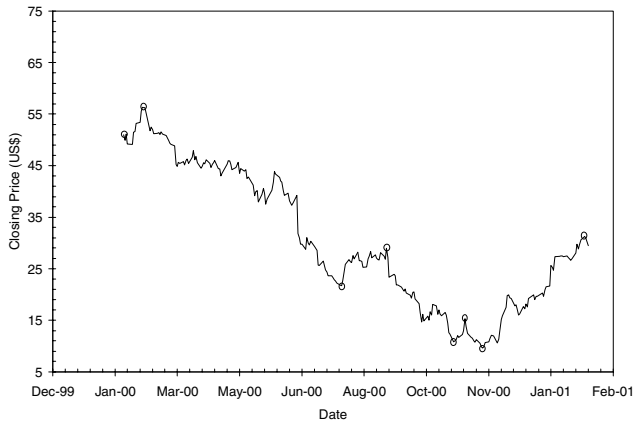

Figure 5: Synthetic Trajectory 3 with Extremal Points

The guiding design criteria for the parameters was to focus on the overall, long term characteristics of the data and discard minor fluctuations.

To specify the parameters to calculate the SLOPE, LENGTH and SNR, a finite impulse response filter (FIR) was used, with N=20 coefficients and a bandwidth B=1/30, which means that cycles with frequencies less than 30 days were removed. N was chosen to provide enough accuracy for our purposes, whereas B was chosen to focus on the major trend. Using the filter with a value of d=30 days and ϵ=0.05, we obtain the slopes shown in Figure 6, which effectively yields the overall trend of the series.

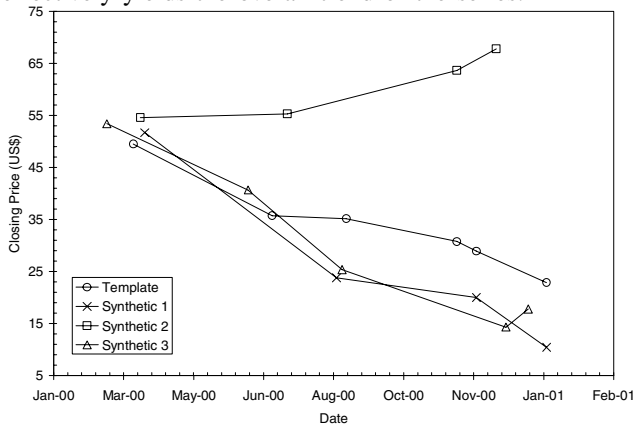

Figure 6: Segments and Slopes

To identify the extremal points a value of R was chosen such that 97% of the points in the template data would be eliminated. This was done so as to focus on major extrema, which in turn offers a view of long term behavior. After a few trials we obtained a value of R=1.3. The extremal points obtained in this way are identified by small circles in Figures 3, 4 and 5. Once the extremals were obtained, we calculated VL and RATIO. Finally, by computing incremental differences between consecutive prices in the series,

Table 1: Feature Vectors for Sample Time Series

Feature	Dataset				Weight
	Template	Synthetic 1	Synthetic 2	Synthetic 3	
SNR(1)	0.1607	0.1389	0.0332	0.0706	2
SNR(2)	0.0679	0.1274	0.0452	0.1903	2
SNR(3)	0.0984	0.1219	0.0368	0.2538	2
LENGTH(1)	68.8	94.3	72.2	69.5	1
LENGTH(2)	36	71.3	85.8	48	1
LENGTH(3)	56	32.4	18.5	80.5	1
SLOPE(1)	-0.2001	-0.2964	0.0109	-0.1860	10
SLOPE(2)	-0.0152	-0.0527	0.0983	-0.3174	10
SLOPE(3)	-0.0788	0.0461	0.2164	-0.1361	10
Kurtosis_Δ	9.3066	6.9049	12.9378	8.4693	1
Skewness_Δ	-0.9209	-1.4493	-1.9330	-0.9199	1
Mean_Δ	-0.0782	-0.1726	-0.0033	-0.0833	1
Ratio(1)	1.0866	1.0585	0.9792	1.1045	2
Ratio(2)	0.5492	0.3156	1.4023	0.3852	2
Ratio(3)	1.3346	1.5221	0.7096	1.3356	2
VL(2)	55.4400	54.0048	49.9600	56.3491	1
VL(3)	30.4500	17.0459	70.0591	21.7082	1

we estimated SKEWNESS_Δ, KURTOSIS_Δ and MEAN_Δ. It was then left to choose the maximum possible number of values n for each feature and collect these in the feature vector. The results of such a procedure are shown in Table 1. In the case of VL, the first point was the same for all samples, and so it was discarded.

5.2 Classifier Design

Given the feature vectors, it was only left to compute similarity between template data and synthetic data. The goal of the study was to design a classifier that would reject synthetic trajectories exhibiting similarity smaller than some specified threshold t. With a suitable threshold the first and third synthetic trajectories shown in the figures would be accepted, while the second would be rejected.

The purpose of introducing weights was to maximize separation between the accept and reject classes. We wanted to give emphasis to the overall trend measured by SLOPE, followed by extrema features such as SNR and RATIO, and finally followed by others that are less controlled by parameters, such as KURTOSIS_Δ, etc. We (heuristically) assigned a weight of 60% to the slopes, 25% to the second set of features and 15% to the rest. Further distributing these percentages between repeated features, we obtained the weights shown in Table 1.

We applied four similarity measures to the feature vectors, in both their weighed and original form. The results are shown in Table 2. In designing the classifier, we were mainly interested in distinguishing between data that was to be accepted or rejected, rather than absolute values of similarity measures. For that reason, similarity measures were ranked according to their ability to draw clear distinctions between the accept and reject values, measured by the degree of separation between them. A high degree of separation also minimizes the possibility of false positives or false negatives.

The values in Table 2 show that the weighed similarity measures have a higher degree of separation, than the basic measures. Likewise, the mean similarity outperfoms other measures, while the peak similarity seems to be relatively weaker. Based on the observed values, we chose the weighed mean similarity for the classifier. We used a threshold of $t = 0.4$, which is midway between the nearest accept and reject values.

6 CONCLUSIONS

We presented a methodology to generate semistructured data realizations based on a template; this can be used to generate random (synthetic) trajectories that are "similar" to a given template. The synthetic data is random, but also exhibits features considered important in the template. Multiple parameters enable an analyst to adjust the model to his particular needs.

We outlined how parameters can be chosen based on overall design criteria; in our case, we focused on the long term. We introduced weighed similarity measures and showed both its ability to focus on certain features, as well as its effectiveness in maximizing separation between datasets that should be either accepted or rejected. The methodology can be applied hierarchicaly, by partitioning data into segments and doing piecewise generation and feature extraction.

Table 2: Similarity Measures between Template and Synthetic Feature Vectors

Similarity Measure	Synthetic 1	Synthetic 2	Synthetic 3
Weighed			
Peak	0.6823	0.5468	0.8028
Mean	0.6864	0.2593	0.5816
Root Peak Square	0.729	0.5816	0.8205
Root Mean Square	0.7604	0.4344	0.6701
Original			
Peak	0.7552	0.6604	0.8168
Mean	0.7015	0.504	0.762
Root Peak Square	0.7952	0.7024	0.851
Root Mean Square	0.7557	0.6095	0.8171

Our results are preliminary and a side-issue in the study of trajectory recognition schemes which exploit higher-order sequences and conditioning to capture and recognize data characteristics. We are currently studying feature extraction and classification algorithms, as well as developing heuristics to choose parameters.

ACKNOWLEDGMENTS

Research supported in part by DoD DAAG55-98-1-0246 and PRF-6903235.

REFERENCES

Aho, A. V., R. Sethi, and J. D. Ullman. 1986. *Compilers*. Boston, MA: Addison-Wesley.

Brookshear, J. G.. 1989. *Theory of computation: Formal languages, automata, and complexity*. Benjamin/Cummings Series in Computer Science. Upper Saddle River, NJ: Pearson Education.

Fink, E. and K. Pratt. 2003. Indexing of compressed time series. In *Data mining in time series databases*,ed. M. Last, A. Kandel, and H. Bunke, 43–65. Singapore:World Scientific.

Friedman, M., and A. Kandel. 1999. *Introduction to pattern recognition: Statistical, structural, neural and fuzzy logic approaches*. Series in Machine Perception and Artifical Intelligence., Vol. 32. Singapore: World Scientific.

Fu, K. S. 1982. *Syntactic pattern recognition and applications*. Englewood Cliffs, NJ:Prentice Hall.

Gavrilov, M., D. Anguelov, P. Indyk, and R. Motwani. 2000. Mining the stock market: Which measure is best ? In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 487–496. New York, NY: ACM Press.

Jurafsky, D., C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 189–192.Piscataway, NJ:Institute of Electrical and Electronics Engineers.

Kearns, M., and L. Ortiz. 2003. The Penn-Lehman automated trading project. *IEEE Intelligent Systems* 18(6):22–31.

Last, M., Y. Klein, and A. Kandel. 2001. Knowledge discovery in time series databases. *IEEE Transactions on Systems , Man , and Cybernetics*, 31B(1):160–169.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. 3d ed. Boston, MA: McGraw-Hill.

Malkiel, B. G. 2004. *A random walk down wall street*. 8th ed. New York, NY:W.W. Norton & Company.

Mitchell, T. 1997. *Machine learning*. Boston, MA: McGraw-Hill.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems : Networks of plausible inference*. San Francisco, CA:Morgan Kaufmann.

Pynadath, D. V., and M. P. Wellman. 1998. Generalized queries on probabilistic context-free grammars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):65–77.

Ramos, J. R., and V. Rego. 2005. Financial data and information. Report in preparation. Technical report CSD, Department of Computer Science, Purdue University.

Sakakibara, Y., M. Brown, R. Underwood, I.S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, 284–283. Honolulu: IEEE Computer Society Press.

Steedman, M. J. 1984. A generative grammar for jazz chord sequences. *Music Perception*, 2(1):52–77.

Stolcke, A. and S. Omohundro. 1994. Inducing probabilistic grammars by bayesian model merging. In *Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, 106–118. London, UK: Springer Verlag.

Sullivan, R., A. Timmermann, and H. White. 1998. Data snooping, technical trading, rule performance and the bootstrap. *Financial Markets Group*, discussion paper 303. London, UK:London School of Economics.

AUTHOR BIOGRAPHIES

JORGE R. RAMOS is a Ph.D. candidate in Computer Sciences at Purdue University. He received a Masters degree in Computer Sciences from Purdue University in 2003. His current research interests include simulation systems, steganography, time series data mining, pattern recognition and machine learning. His e-mail address is <jrramos@cs.purdue.edu>.

VERNON REGO is a Professor of Computer Sciences at Purdue University. He received his M.Sc.(Hons) in Mathematics from B.I.T.S. (Pilani), and an M.S. and Ph.D. in Computer Science from Michigan State University (East Lansing) in 1985. He was awarded the 1992 IEEE/Gordon Bell Prize in parallel processing research, and a 1988 DFG German Research Council Network Research Award. His research interests include parallel & stochastic simulation, probability modeling, distributed computing, software, and financial engineering. His e-mail address is <rego@cs.purdue.edu>.