PREDICTING CHORDAL TRANSITIONS IN JAZZ: The Good, the Bad, and the Ugly

Belinda Thom
bthom@cs.cmu.edu
Department of Computer Science
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213

Abstract

Instance-based machine learning techniques have been applied to predict the next chord in a piece of jazz music. This study compares off-line (trained on a static data base of songs) and on-line (trained on-the-fly) prediction techniques. A hybrid model combines both on-line and off-line models. On average, chords in a jazz piece are predictable 53% ± 17% of the time using the hybrid model. Successful predictions were defined as exact matches of chord root and type.

1 Introduction

How well can one anticipate the next chord in a piece of jazz music? A sampling of seasoned musicians, at best, will give you a qualitative answer. Predicting chords in a piece of jazz music is a *time series prediction* problem [Weigend and Gershenfeld, 1994]. Reliable prediction requires the underlying process to adhere to some well defined behavior. Assuming the behavior is a function of underlying state, the problem is to infer behavioral patterns from observable data. Consequently, prediction is limited to the extent the forces driving a process obey some structure. Learning algorithms are further limited by how much underlying state is available in the training data [Rabiner, 1989; Sholom and Kulikowski, 1991].

1.1 Key Issues

Finding an adequate software representation for the chord prediction problem is crucial. Rather than using knowledge-based methods, harmonic structure was directly inferred from a large data base of jazz chord progressions. Given this data-driven approach, an adequate problem specification must answer the following questions:

- What features determine harmonic progression in jazz?
- What encoding optimizes the ability to generalize from individual training examples?
- What algorithms both span the concept to be learned and optimize convergence towards this concept?
- What training set best represents the universe of jazz chord progressions?
- How can the system be tested in order to bound performance on the universe as a whole?

2 Restrictions Imposed by Jazz Chord Prediction

Predicting chords in jazz limits the applicability of many machine learning algorithms. In addition, this problem adds a few twists to the task of selecting data and measuring performance.

2.1 Chords are Ordinal

The lack of a continuous "similarity measure" between every pair of chords is a serious handicap. Not only does this lack preclude the use of neural network, function approximation, and nearest neighbor techniques, it also hinders discrete learning algorithms because their only form of feedback is dichotomous: "exact match" or "terrible error".

2.2 Algorithms Must Perform Incremental Learning

Incremental learning is absolutely necessary given the belief that the concept driving a progression changes within a song and among different songs. Incremental learning must also be very fast because the computer should not miss many chords in a row. Unfortunately, the lack of continuous feedback also hinders an algorithm's ability to adapt responsively.

2.3 Selecting an Appropriate Training Set is Problematic

Typical machine learning scenarios use randomly sampled training and test sets [Sholom and Kulikowski, 1991]. It is assumed that the training and test set distributions directly represent the universe distribution. The purpose of the test set is to determine when to stop learning[1] and validate the performance of the learned concept [Sholom and Kulikowski, 1991]. Because it only makes sense to measure performance with respect to an entire song, test and training sets should be made up of *all* the chords in randomly selected jazz *songs* (see Figure 1). Per song selection is problematic because distributions within them are highly non-random. The predictive performance observed for one song is a poor indicator of another song's expected performance. Test and training sets may be very different from each other and from the universe. Finally, it is unclear how to normalize songs so all receive equal representation. Clearly, this type of *batch learning* must involve adaptation and adds an additional level of ambiguity to the meaning of *expected value*.

[1] To prevent over-fitting.

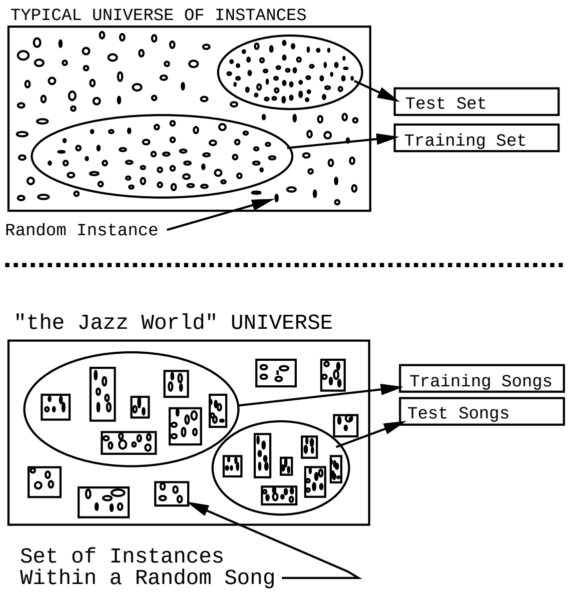

Figure 1: Machine Learning Environments

2.4 Measuring Performance

Evaluating a predictor's performance is difficult because the impact its errors have on a song is a function of previous errors, the temporality of the error (and previous errors), the context of the song, etc. For example, what is worse: errors that occur evenly spaced in time or errors that occur in clusters[2]? Often when adaptive algorithms are used, there is a significant amount of data available for on-line learning (e.g. text compression), and average performance is an acceptable measure of an algorithm's success. In jazz, an average song may contain thirty unique chords and song lengths vary significantly[3]. In this scenario, average values are misleading for several reasons:

- An average metric does not address temporality or context.
- The *Average Number of Incorrect Chords Per Song* metric does not take into account song length.
- The *Percentage Incorrect Per Song* metric exaggerates the error seen in shorter songs and the resolution of this metric is a function of song length (as far as resolution is concerned, all songs are too short).

In this study, consequently, it was difficult to select a statistically significant *best* predictor for a given song because often average metrics for a song's better predictors differed by, at most, a chord or two.

[2]Imagine a scenario where a burst of errors may be less intrusive (e.g. during a one-bar rhythmic progression); unfortunately, this type of reasoning requires high-level musical understanding, which the computer does not possess.

[3]This definition of song length is the number of *unique chords* once through, without repeats.

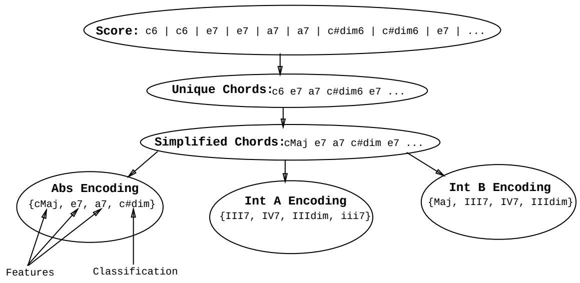

Figure 2: Reducing a Score to Training Instances (the 1st instance generated for each encoding is shown); n=3.

3 Problem Representation

The problem was restricted to predicting the next unique chord, given a chord progression history and tonic[4]. Each of n most recent chords was a feature. These features were used to classify the next chord (both pitch and type). Several feature sets were considered (ranging from no history to three most recent chords). To encourage generalization, chord types were mapped to the restricted set:

- Major
- Dominant
- Minor
- Diminished
- Augmented
- Unknown

and Minor 6th chords were mapped to their Major equivalent. The following encodings were considered:

- *Absolute (Abs)*: chord histories were referenced to their absolute pitch.
- *Interval (Int A)*: chord histories were referenced with respect to previous pitch.
- *Interval (Int B)*: same as Int A, except that the earliest feature is only chord type.

Figure 2 shows a reduction from score to training instances for each encoding. In this example, a history window of size three (n=3) is used. For the interval encodings, major and minor intervals are specified using the standard notation[5]. Notice that the Int A encoding needs an initial reference pitch and, therefore, its first feature occurs one chord later than the Abs and Int B encodings.

The jazz chord prediction problem naturally lends itself to the *n-gram models* used by the natural language community [Bell *et al.*, 1990]. Learning algorithms of this sort (simple instance-based classification) work on ordinal domains and ranges, support incremental learning, and optimize performance by maximizing likelihood. In particular, a prediction is determined by [Bell *et al.*, 1990]:

$$Argmax_{i \in Chords} = Pr(i|f_{t+n}, \ldots, f_t) = \frac{Pr(i, f_{t+n}, \ldots, f_t)}{Pr(f_{t+n}, \ldots, f_t)}$$

[4]All songs were transposed to the key of C.

[5]For example, III is a major interval and iii is a minor interval.

where f_t is the least recent feature and f_{t+n} is the most recent feature in the history window.

The ultimate goal is to create an expert jazz predictor that can adapt to the vagaries of a song's local structure. Using n-grams, this goal can be accomplished as follows. Initially, when nothing is known about a particular song, it is reasonable to consult the training set for these probabilities. As data for a song becomes available, these probabilities should be adjusted accordingly. Using this scheme, predictive performance is intimately tied to:

- How well the underlying harmonic structure is captured by small, finite context models.
- How well the training set covers the song being predicted.
- To what degree a song's on-line training examples cover itself (i.e. how repetitive is it?).

As mentioned earlier, complications exist that are specific to this time series prediction problem. Because a thorough investigation of these complications are beyond the scope of this study, many of these issues were avoided with the following simplifications. Aside from the modifications used to generate the training examples, performance was measured with respect to exact matches. An exact match of chord pitch and type was correctly classified; all other classifications were equally wrong. Fifty songs selected at random from a jazz fake book were used to generate the training and tests sets. All superfluous repeats within a song were removed. Cross-validation techniques were used to measure the predictive accuracy of each song individually because an average of these tests provides a reasonable expectation of per song performance [Sholom and Kulikowski, 1991]. Bear in mind that a particular song in the universe could do much better (or much worse) than this average, depending upon how much repetition the song contains and how much it differs from the maximum likelihood as defined by the training set. Finally, per song averages were reported versus song length in order to provide an indication of responsiveness.

4 Representation Performance

Experiments were run to determine how well the feature sets and encodings capture musical structure by looking at the *best possible case*. In this scenario, individual songs were used as both the training and test sets. Off-line models for each encoding and window size were trained on a given song. These models were then used to predict the song itself. It is surprising how much structure is captured with the reduced musical representation. As one would expect, larger context models (i.e. higher order, larger history window) reduce the degree of overlap among training examples with different classifications, resulting in lower entropy models and better predictions [Bell *et al.*, 1990]. As shown in Figure 3, the Abs encoding provides lowest entropy per song, on average. For some songs, Int A encodings provide better separation of classifiers than Abs encodings. Int B always under-performs given the same history window size, but this is an unfair comparison because one of its features only contains chord type. Although highest order (n=3) provides optimum performance for all songs, optimal encoding is song dependent. Figure 4 displays the performance of 3^{rd} order models for each song and each encoding; given an optimal encoding, songs were uniquely determined 92% ± 4% of the time, on average.

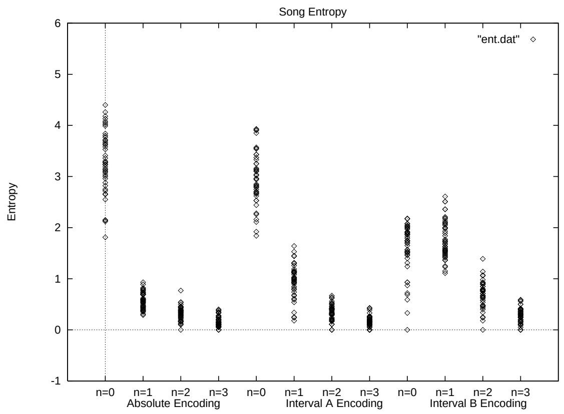

Figure 3: Song Entropy as a Function of History Window and Encoding Type

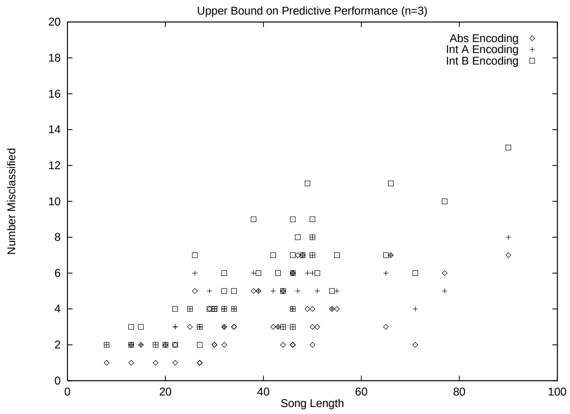

Figure 4: Predictive Performance: Model is Trained Exclusively on the Song it is Predicting

Although these experiments do not answer the prediction question (predicting a song already seen is trivial!), they do provide a wealth of information. The entropy of each model is the entropy of the song itself[6], and on/off-line predictors are ultimately limited by the degree to which they replicate the structure captured by these models. These results indicate there is no universal, optimal encoding for songs. Furthermore, it is surprising the Abs encoding performed better (on average), given jazz has such a large intervalic harmonic component. *Although intervals are important, many songs contain enough non-intervalic structure that Abs allows better overall segregation.* Bear in mind that each encoding maps both features and classification with respect to its reference. Identical features are then grouped together; model entropy reflects how many unique classifier encodings result in each group. It just so happens that intervalic representation causes more training instances within a song to collide with one another.

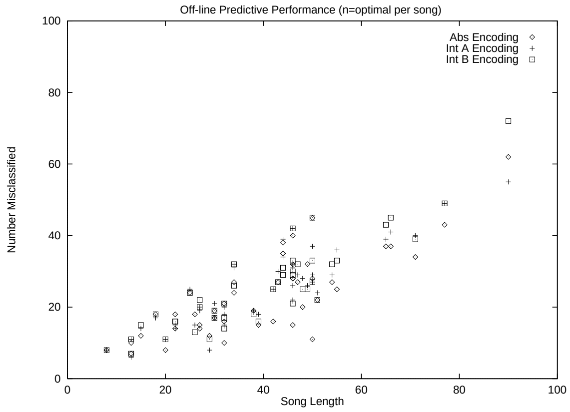

Figure 5: Predictive Performance: Model is Trained on Every Song but the Song it is Predicting

5 Off-line Prediction

Experiments were run to investigate two questions:

- How much harmonic similarity exists among jazz pieces?
- To what degree can we extract appropriate generalizations given this representation scheme?

Each song was a test set exactly once. It was used to test the performance of off-line models trained using the remaining songs (models for each encoding and window size were built). The results of these tests were both surprising and unfortunate. Both optimal encoding and window length were song dependent. This fact makes the matter of selecting a global expert predictor all the more difficult. Perhaps more worrisome, no window size or encoding scheme provided a clear advantage when considering the entire set of songs[7]. Although one would expect coverage to decrease as context increases, the critical issue is one of over-fitting; larger context models tended to "memorize" particular chord phrases [Bell *et al.*, 1990]. In particular, 3^{rd} order models were very sparse, lacking enough data to infer statistically significant trends beyond the most commonly occurring phrases (e.g. iim-V7-IMaj, etc.). It is surprising, however, that given a wealth of forty-nine jazz pieces, per song predictive performance is significantly degraded and erratic: for each song's optimal encoding and window size, predictive accuracy was $42\% \pm 17\%$. Results for each song's optimal window length are shown in Figure 5.

Clearly, harmonic similarity exists between jazz songs; however our representation yields a performance of less than half the upper bound! I attribute the batch problem described earlier as a major cause of this degradation. When several significant trends conflict (e.g. V7-im-iidim, V7-im-iim), only the most likely is predicted. This situation is a serious problem because many songs might be exclusively centered around a trend that is less likely with respect to the entire training set average (or the entire universe average). In addition, prediction is based on matching each feature exactly. This requirement significantly hinders the ability of the computer's algorithm to generalize, since classification cannot infer different rules for different contexts (ideally, different subsets of features would be used to classify different situations). In order to address this point, a brief experiment with standard decision tree techniques[8] [Quinlan, 1993] was conducted (with and without pruning); unfortunately, little improvement was observed. These experiments indicate that structure exists among songs, but each song has its own identity and contains a strong and unique local structure. *Adaptable models are absolutely necessary.*

6 On-line Prediction

Experiments were run to investigate the feasibility of on-the-fly learning, given the per song scenario. Performance is intimately tied to the degree of repetition within each song. Each song was its own test set. On-line models began with no knowledge; as they were tested, a database was built up incrementally. Models for each encoding and window size were built. For these tests, higher order models always performed best, but optimal encoding was still song dependent. It is surprising that, given no information, average per song predictive performance is on par with off-line prediction because it implies learning over a short sequence of chords is more powerful than consulting a carefully constructed data base. Given the best encoding for each song, average performance was $49\% \pm 15\%$. Results are shown in Figure 6.

Initially, local prediction is completely random. Over a short span (e.g. thirty chords), prediction improves to the point that off-line prediction is outperformed. (Remember performance is compared by means of an average metric that does not even take into account the temporal aspect of the errors!) In addition, almost all errors in this paradigm occurred because an instance was not covered; when 2^{nd} and 3^{rd} order models predict, they usually predict correctly. As a result of these on-line prediction experiments, I believe small amounts of data from the current song contain a lot more statistical significance than that of an entire compilation of songs.

[6] Relative to the order of the model and the encoding.

[7] The distribution of optimal window length and encoding scheme across all fifty songs appeared random.

[8] These techniques use greedy strategies to select subsets of features in order to attempt to minimize the entropy of each classification.

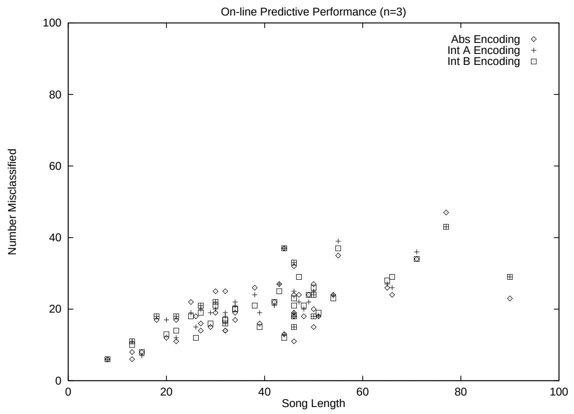

Figure 6: Predictive Performance: Model Learns Everything On-The-Fly

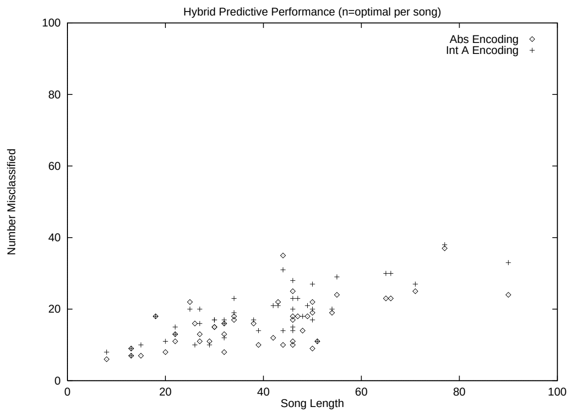

Figure 7: Predictive Performance: Model Initialized with Off-line Data and Learns On-The-Fly

7 Combining Models: The Best of Both Worlds

It is non-trivial to combine various models into a global predictor that is superior over the large domain of jazz pieces. Most n-gram modeling schemes use some form of smoothing to combine models of various order [Bell *et al.*, 1990]. These schemes usually prefer models of higher order by giving them significantly higher weights in combined probability density functions than their lower order counterparts[9]. As described earlier, with respect to off-line predictors, smoothing does not eliminate the effects of over-fitting and does not guarantee improved performance with higher order. Although ad-hoc methods exist for smoothing multi-order models by conditioning them using data from a separate test set [Bahl *et al.*, 1983], there is still the issue of selecting this set in such a way that it generates a representative result. Alternatively, one could keep each predictor separate, selecting the best one adaptively in real-time [Blum, 1995]. The issue becomes determining what predictor is superior right now. This task is made more difficult by short sequences of chords in a song and the close overall performance of several predictors. For this reason, common temporally adaptive schemes (e.g. Weighted Majority, Winnow [Littlestone, 1987]) become less attractive. Experiments using these adaptive methods produced global predictors that often approached optimal performance within two or three chords; however, the statistical significance of deviations of this magnitude is questionable. In fact, for a given song, magnitudes of this size were often what discriminated between optimal and sub-optimal encoding schemes and window sizes.

These types of issues arise whether attempting to optimize performance via smoothing or temporal adaptation. Many ad-hoc methods have been tried for improving performance. Of these, the methods which prove optimal appear to be song dependent. *Smoothing and adaptation are messy issues in general!* This observation is especially true for the case of jazz chord prediction, where short song lengths and highly local structure make the statistical validity of patterns observed in the data more questionable. In this study, these issues were avoided by using simplistic smoothing and adaptive techniques. For all experiments, models of different order were smoothed by assigning weights as a function of frequency alone [Bell *et al.*, 1990]. If all things were equal, larger order models were preferred. Adaptation was implemented as follows. Initially, off-line models were consulted for likelihood estimates. As on-line data became available, it was urgently incorporated into the predictor by weighting it fifty times more heavily than a single off-line instance. This adaptive model is termed the *hybrid* model.

[9]This scheme assumes that more context implies a more reliable estimate. Lower order models are consulted when higher order models do not cover a training example or when selection must occur among several statistically similar events in higher order models.

8 Hybrid Prediction

Experiments were run to investigate the performance of the hybrid approach within the per song framework. Performance is intimately tied to the likelihood of a song, given the off-line database and how quickly a song's repetitive structure becomes most likely. Each song was its own test set. Off-line models were built from the remaining songs for each encoding and window size. For these tests, longer context usually performed best, but optimal encoding was still song dependent. As one would expect, performance was better than for on-line or off-line predictors alone. Given the optimal encoding for each song, average performance was 53% $\pm$ 17%. Results are shown in Figure 7.

9 Future Research

A primary result of this study is the number of questions it raises with regard to inferring harmonic structure from encoded jazz scores. These questions include:

- What is the best way to measure the distance between ordinal chord values? Although in many cases, inexact predictions were musically inappropriate, in others, they were quite tasteful. Furthermore, it is desirable to reward (rather than penalize) predictors that perform tasteful chord substitution. After all, this is a fundamental element of jazz! This issue also raises an important philosophical question: *What is the difference between predicting the next chord and generating a reasonable chord substitution?*

- What constitutes a reasonable training set? How does training set effect performance? Would multiple databases (tuned to particular styles) perform much better? How critical is the size of the training set? Can style based databases be used to classify particular jazz styles?
- How can domain knowledge be incorporated to improve performance? Can musical knowledge be incorporated into maximum likelihood calculations? How can domain knowledge be incorporated into temporally adaptive methods?
- Are there less rigid ways to encode past history so the essence of jazz chord structure becomes more general and predictable? What methods are available for generating instances based on a partial match of features?

10 Conclusion

The most useful result of this study is what it says about the similarities that arise in jazz chord progressions. Although patterns are shared among songs, on average, local data seems to provide much more reliable predictive information. This study is also useful because it bounds the level of performance one can expect when restricted by the exact match constraint. The results of this study indicate that computer jazz chord prediction succeeds because songs have highly repetitive, localized chord structure. This observation offers many interesting applications. For example, in what ways can we employ computer chord predictions to enhance improvisational performance? How well can a computer accompany free-form improvisation? Can predictive variables (e.g. an expert's local and/or long term performance) be used to enhance algorithmic composition and/or video generation? Ultimately, the essence of jazz performance is interactive and improvisational. Improving the computer's ability to anticipate chords and understand chord structure is critical for its active participation in this art form.

References

[Bahl *et al.*, 1983] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–19, March, 1983.

[Bell *et al.*, 1990] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression.* Prentice Hall Advanced Reference Series, Prentice Hall, New Jersey, 1990.

[Blum, 1995] Avrim Blum. Empirical Support for Winnow and Weighted-Majority based Algorithms: Results on a Calendar Scheduling Domain. To be appearing in *Machine Learning 1995*.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1993.

[Littlestone, 1987] Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear Threshold Algorithm. In *Machine Learning II,* pages 285-318, Kluwer Academic Publishers, Boston, Massachusettes, 1987.

[Rabiner, 1989] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, February, 1989.

[Sholom and Kulikowski, 1991] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[Weigend and Gershenfeld, 1994] Andreas S. Weigend and Niel A. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past: Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis.* Addison-Wesley Publishing Co., Reading, Massachusettes, 1994.