Using Dynamic Time Warping to Find Patterns in Time Series

Donald J. Berndt
James Clifford
Information Systems Department
Stern School of Business
New York University
44 West 4th Street
New York, New York 10012-1126
{dberndt, jclifford}@stern.nyu.edu

April 26, 1994

Abstract

Knowledge discovery in databases presents many interesting challenges within the context of providing computer tools for exploring large data archives. Electronic data repositories are growing quickly and contain data from commercial, scientific, and other domains. Much of this data is inherently temporal, such as stock prices or NASA telemetry data. Detecting patterns in such data streams or time series is an important knowledge discovery task. This paper describes some preliminary experiments with a dynamic programming approach to the problem. The pattern detection algorithm is based on the dynamic time warping technique used in the speech recognition field.

Keywords: dynamic programming, dynamic time warping, knowledge discovery, pattern analysis, time series.

1 Introduction

Almost every business transaction, from a stock trade to a supermarket purchase, is recorded by computer. In the scientific domain, the Human Genome Project is creating a database of every human genetic sequence. NASA observation satellites can generate data at the rate of a terabyte per day. Overall, it has been estimated that the world data supply doubles every 20 months [FPSM91]. Many databases, whether formed from streams of stock prices, or NASA telemetry, or patient monitors, are inherently temporal. The challenge of knowledge discovery research is to develop methods for extracting valuable information from these huge repositories of data—most of which will remain unseen by human eyes.

The data on which our knowledge discovery tools operate can be characterized by the traditional distinction between *categorical* data and *continuous* data [Ker86]. Categorical data are characterized by nominal scales and simple group membership. An example pattern based on categorical data would be "most residents of Manhattan are Democrats". Continuous data may be ranked on ordinal scales, and differences become meaningful on interval and ratio scales. Queries on continuous data are more complex since they include relationships such as "less than" and aggregations such as averages or maximums. For instance, we might find patterns such as "the average salary of Democrats that subscribe to the New Yorker is greater than $50,000".

Both categorical and continuous data can be extended by adding a temporal dimension—creating *time series* data. The last decade has witnessed a tremendous growth of interest and research in the field of temporal databases, as illustrated in [TCG+93]. Among the temporal data models that have been proposed in the literature, some ([CC87]) directly model time-series data; the advantages of these models have recently been clarified in [CCT94]. But before we can use the results of queries on time series, we must be able to detect temporal patterns such as "rising interest rates", or the small pattern in Figure 6, in the underlying time series data. This is an interesting sub-problem within the context of discovering higher-level relationships and is the focus of this preliminary paper. The detection of patterns in time series requires an approximate or "fuzzy" matching process. These patterns may then be used to construct higher-level rules such as "Democrats with steeply rising salaries are likely to subscribe to the New Yorker".

For example, consider the line graphs in Figure 1 depicting lynx and snowshoe hare populations over nearly a century [Odu71] [Cla73].[1] What type of patterns emerge from the data? One pattern seems to be that the "lynx population rises after an increase in the snowshoe hare population"—fairly natural given the predator/prey relationship. In addition, the "lynx population appears to be less volatile than the snowshoe hare population". Lastly, there are two "spikes" in the hare population that might be explained by a third data stream—perhaps they are associated with "fashion sense" as tracked by sales of rabbit fur garments! Of course, Figure 1 might represent data collected from any domain (just replace the labels). One could imagine the hare population to be data tracking a computer industry stock index and the lynx population to be a particular issue being "pulled" by movements of the index.

Humans are very good at visually detecting such patterns, but programming machines to do the same is a difficult problem. The difficulty arises in capturing the ability to match patterns with some notion of "fuzziness". Similar approximate pattern detection tasks are found in several fields. Related work in statistics [JD88], signal processing [Poo88], genetic algorithms [Gol89] [Pac90], and speech recognition[Ain88] offer a variety of useful techniques.

[1] The ratio of predator to prey seems to large, but the data are derived from commercial trapping, not from scientific observation.

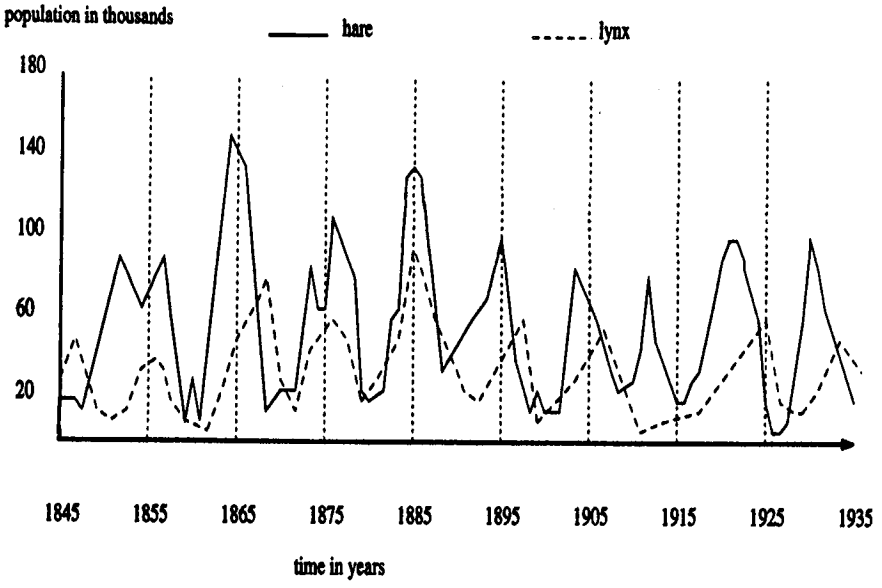

Figure 1: Changes in the Lynx and Snowshoe Hare Populations as Determined by the Number of Pelts from the Hudson's Bay Company.

2 Dynamic Time Warping

In particular, the problem of recognizing words in continuous human speech seems to include many of the important aspects of pattern detection in time series. Word recognition is usually based on matching word templates against a waveform of continuous speech, converted into a discrete time series. Successful recognition strategies are based on the ability to approximately match words in spite of wide variations in timing and pronunciation. Recently, speech recognition researchers have used dynamic programming as the basis for isolated and connected word recognition [Ain88] [RL90] [SC90].

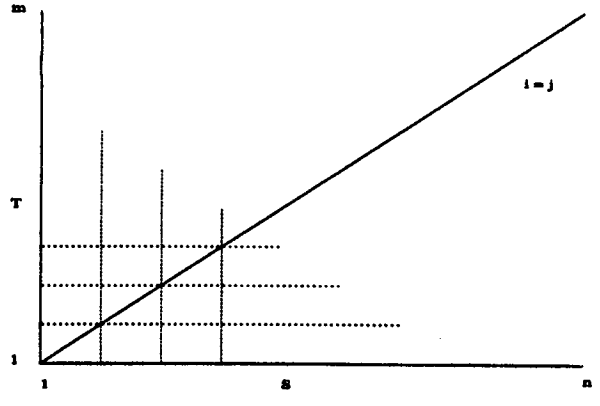

Figure 2: An n by m Grid.

The technique of *dynamic time warping* (DTW) uses a dynamic programming approach

to align the time series and a specific word template so that some distance measure is minimized. Since the time axis is stretched (or compressed) to achieve a reasonable fit, a template may match a wide variety of actual time series. Specifically, the pattern detection task involves searching a time series, S, for instances of a template, T.

$$S = s_1, s_2, \ldots, s_i, \ldots s_n \tag{1}$$

$$T = t_1, t_2, \ldots, t_j, \ldots t_m \tag{2}$$

The sequences S and T can be arranged to form a n-by-m plane or grid (see Figure 2), where each grid point, (i, j), corresponds to an alignment between elements s_i and t_j. A *warping path*, W, maps or aligns the elements of S and T, such that the "distance" between them is minimized.

$$W = w_1, w_2, \ldots, w_k \tag{3}$$

That is, W is a sequence of grid points, where each w_k corresponds to a point $(i, j)_k$, as shown in Figure 3.[2] For example, point w_3 in Figure 3 indicates that s_2 is aligned with t_3.

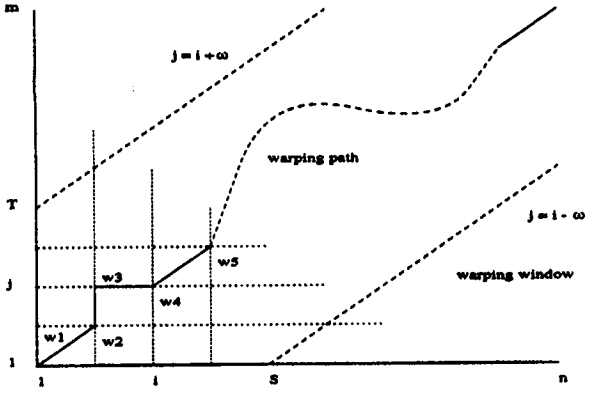

Figure 3: An Example Warping Path.

In order to formulate a dynamic programming problem, we must have a distance measure between two elements. Many distance measures are possible—two candidates for a distance function, δ, are the magnitude of the difference or the square of the difference.[3]

$$\delta(i, j) = \mid s_i - t_j \mid \tag{4}$$

$$\delta(i, j) = (s_i - t_j)^2 \tag{5}$$

Once a distance measure is defined, we can formally define the dynamic time warping problem as a minimization over potential warping paths based on the cumulative distance for each path, where δ is a distance measure between two time series elements.

$$DTW(S,T) = min_W[\sum_{k=1}^{p} \delta(w_k)] \tag{6}$$

[2]When there is no timing difference the warping path coincides with the diagonal line, $i = j$.

[3]A given distance measure allows warping paths to be ranked. It may also be useful to define relative (i.e. percentage based) distance measures to make individual distances more comparable. For example, consider comparing distance between stock prices of \$8 and \$10 with that of \$98 and \$100.

In dynamic programming formulations, we need a *stage variable*, *state variables*, and *decision variables* that describe legal state transitions [LC78]. The stage variable imposes a monotonic order on events and is simply time in our formulation. The state variables are the individual points on the grid as illustrated in Figure 2. The decision variables are less easily recognized in our formulation, but correspond to the restrictions on permissible paths between two grid points. These restrictions serve to reduce the search space—the space of possible warping paths. Searching through all possible warping paths is combinatorially explosive. Therefore, out of concern for efficiency, it is important to restrict the the space of possible warping paths—some restrictions are outlined below [SC90].

1. **monotonicity**
   The points must be monotonically ordered with respect to time, $i_{k-1} \leq i_k$ and $j_{k-1} \leq j_k$.

2. **continuity**
   The steps in the grid are confined to neighboring points, $i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$.

3. **warping window**
   Allowable points can be constrained to fall within a given warping window, $| i_k - j_k | \leq \omega$, where ω is a positive integer window width (see Figure 3).

4. **slope constraint**
   Allowable warping paths can be constrained by restricting the slope, thereby avoiding excessively large movements in a single direction.

5. **boundary conditions**
   Lastly, boundary conditions further restrict the search space. The most restrictive variant uses constrained endpoints, such as $i_1 = 1$, $j_1 = 1$ and $i_k = n$, $j_k = m$. Rather than simply anchoring the points, we can relax the endpoint constraint by introducing an "offset" in the above conditions. Lastly, a starting point may be specified, with subsequent path constraints replacing a fixed ending point.

The dynamic programming formulation is based on the following recurrence relation, which defines the cumulative distance, $\gamma(i,j)$, for each point.

$$\gamma(i,j) = \delta(i,j) + min[\gamma(i-1,j), \gamma(i-1,j-1), \gamma(i,j-1)] \quad (7)$$

That is, the cumulative distance is the sum of the distance between current elements (specified by a point) and the minimum of the cumulative distances of the neighboring points. The above formulation is a *symmetric* algorithm since both predecessor points off the diagonal are used.[4] The dynamic programming algorithm fills in a table of cumulative distances as the computation proceeds, removing the need to re-calculate partial path distances. Upon completion, the optimal warping path can be found by tracing backward in the table—choosing the previous points with the lowest cumulative distance.

[4] An asymmetric formulation would use only one of the points, $(i-1, j)$ or $(i, j-1)$. Experimental results suggest that symmetric algorithms perform better in the speech recognition domain [SC90].

2.1 Measures of Fit

Once the best warping path is found, a score describing the "fit" of the template and underlying time series segment must be calculated. The score is intended to quantify the degree of fit achievable by stretching or compressing the series and template with regard to time. In addition, the score must allow comparisons so that matches can be ranked. We might be interested in comparisons of multiple matches of a single template against a long series, as well as in comparisons of the fit of various templates with a series.

One possible measure is based on the ratio of the cumulative warping path distance to a baseline distance. We have experimented with several baseline calculations, including a boundary around the time series segment. An example of such a boundary is shown in Figure 5. In this example, the baseline is a relative band centered around the series. For instance, a data point of 50 would be bounded by 25 and 75, assuming the band width to be 100% of the data value. The sum of these boundary intervals makes a reasonable baseline, with the score set to zero if the warping path distance exceeds the baseline (thereby keeping the score in the $[0, 1]$ range).

3 Simple Experiments

In order to illustrate the approach, we use the simple "mountain-shaped" test patterns shown in Figure 4. The collection starts with a mountain that increases in increments of 20, with subsequent patterns "flattened" until we reach a horizontal line at 40.[5]

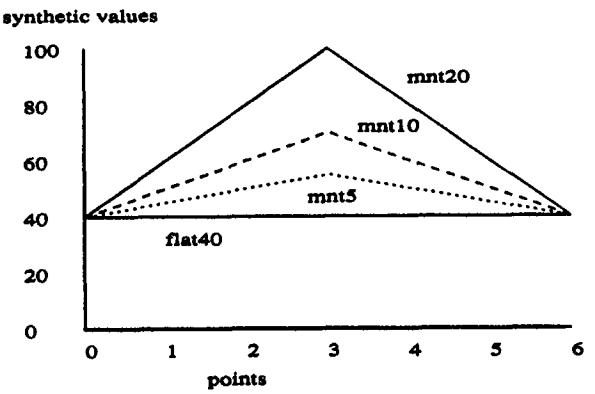

Figure 4: A Collection of Mountain-Shaped Patterns.

We ran a naïve DTW algorithm on all combinations of mountain patterns. That is, each pattern acted as both a series and a template during the matching process.[6] The naïve algorithm uses fixed endpoints and some simple constraints on warping path slope (including the monotonicity and continuity restrictions). The resulting scores are summarized in Table 1.

[5]The units in this example have no particular meaning.

[6]The values on either side of the diagonal are different due to the baseline calculation being dependent on which pattern acts as the series.

scores *template/series*	flat40	mnt5	mnt10	mnt20
flat40	1.00	0.86	0.76	0.61
mnt5	0.84	1.00	0.91	0.73
mnt10	0.68	0.89	1.00	0.85
mnt20	0.36	0.62	0.81	1.00

Table 1: Score Matrix for the "Mountain" Templates.

For example, consider the matching process with *mnt*20 acting as the underlying series and *mnt*10 the specific template. The overall score is 0.85, with the corresponding matrix of cumulative distances (γ) shown in Table 2. The warping path representing the reasonably close fit is shown below, with the appropriate Table 2 cells highlighted.

$$(6,6)(5,5)(5,4)(4,3)(3,3)(2,3)(1,2)(0,1)(0,0) \quad (8)$$

The warping path is a sequence of (i,j) pairs, indicating an alignment between a series element, s_i, and template element, t_j.

6	90	50	70	110	130	90	**70**
5	90	30	50	90	90	**70**	70
4	80	20	40	60	70	**60**	80
3	60	20	**20**	**50**	**60**	70	100
2	30	**10**	30	70	90	90	110
1	**10**	10	40	90	120	130	140
0	**0**	20	60	120	160	180	180
mnt10/mnt20 **distances** (γ)	0	1	2	3	4	5	6

Table 2: Cumulative Distance Matrix for *mnt*20 and *mnt*10.

4 Predator/Prey Experiments

Consider again the time series in Figure 1. The hare population exhibits two "double-topped" peaks which will form the basis for our pattern detection. These peaks may indicate some interesting phenomena, perhaps heavy lynx hunting for pelts caused a break in the normal spike-shaped cycles. In the entirely different domain of stock market technical analysis, involving the search for many such patterns, this particular form is called a "double top reversal" [LR78], Figure 5 is a close-up of the peak occurring near 1850, including the boundary which would be used to calculate a baseline score as discussed in Section 2.1.

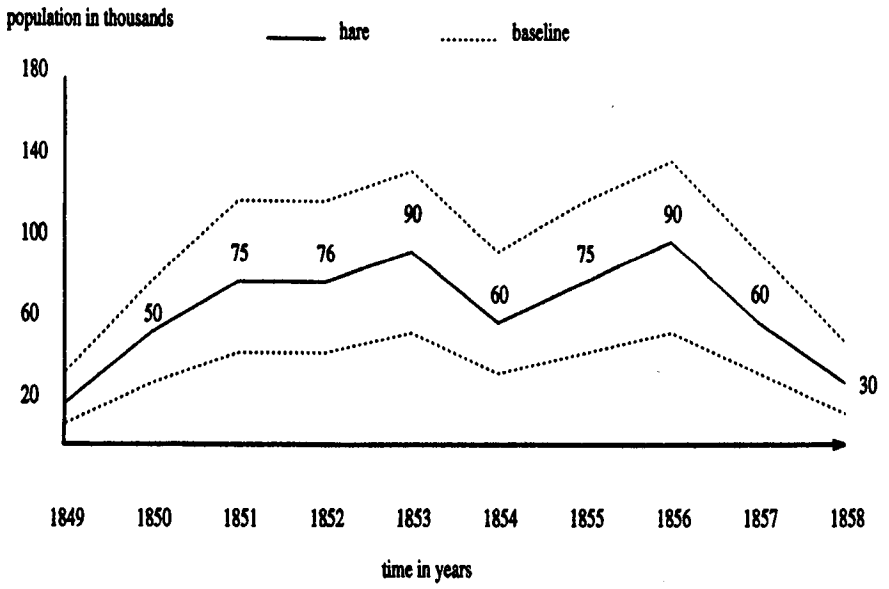

Figure 5: Double Top Peak in the Hare Population Near 1850.

Figure 6 describes an example template for a double top peak. In our experiment, the template is specified in relative terms with respect to the first point. This point serves as an anchor during the matching process, inheriting the starting value from the time series segment being considered. For instance, an anchor value of 20 from the underlying time series will result in a template instance of $20, 40, 60, 80, 70, 70, 80, 60, 40, 20$, as illustrated in the following table:

pattern scaling	0	1	2	3	2.5	2.5	3	2	1	0
template instance	20	40	60	80	70	70	80	60	40	20

After matching the template along the hare population series, again using the DTW algorithm described in Section 3, the two highest scores were anchored at 1849 (0.87) and 1870 (0.86). These points correspond nicely with the double-topped peaks in Figure 1. Most other scores were substantially lower, with only a few reaching over 0.6. This matching problem is fairly difficult since there are several areas which might be classified as double-peaked and the template is quite simple. A more complex template should be harder to match closely, and therefore be more selective. The graph of the scores over the hare population data is shown in Figure 7. The particular warping path anchored near 1850 is shown in Figure 8, with the warping path alignments represented by dotted lines.

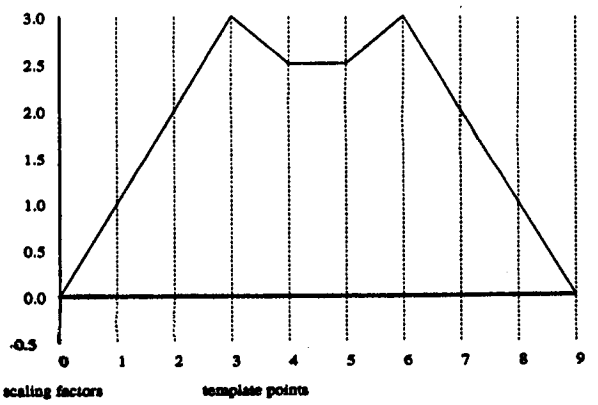

Figure 6: An Example Double Top Template Expressed in Relative Terms.

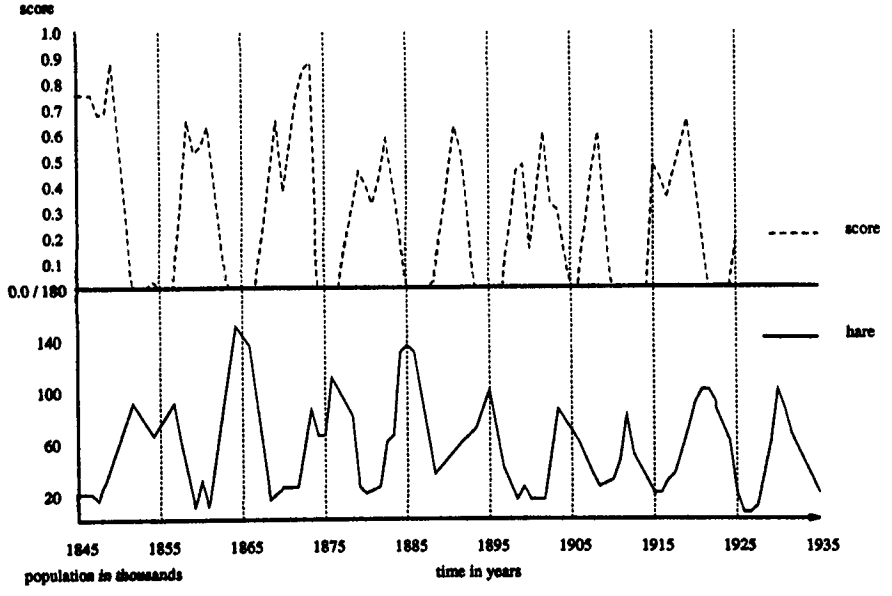

Figure 7: Dynamic Time Warping Scores and Hare Population Data.

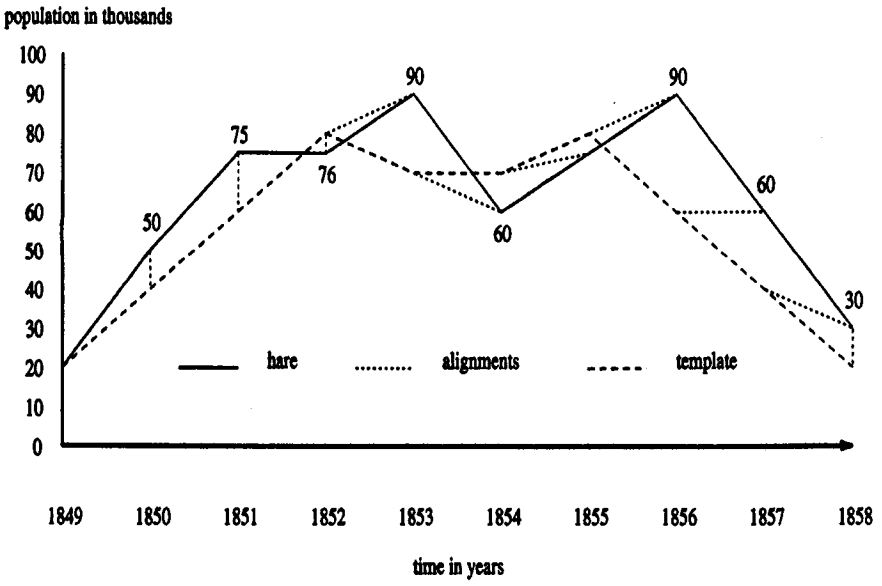

Figure 8: The Series/Template Alignment Anchored Near 1850.

5 Conclusion

The knowledge discovery sub-problem of finding patterns in time series data is a challenging research issue. It is certainly important if we consider how much data is inherently temporal, such as stock prices, patient information, or credit card transactions. Our preliminary experiments with techniques based on dynamic time warping are encouraging.

Once we can detect patterns such as a double top peak, we can express higher-level relationships or "knowledge" as rules, For example, the following rule expresses one possible relationship, where $t1$ and $t2$ represent a time interval.

$$double\_top(hare, t1, t2) \rightarrow heavy\_hunting(lynx, t1, t2) \quad (9)$$

We plan to develop more refined DTW algorithms and evaluate them in the context of a prototype knowledge discovery tool. In addition, we are interested in exploring parallel algorithms to improve performance. Since the DTW algorithm is based on independent matches of a template against segments of a time series, we can divide the time series and use parallel matching processes. The granularity of the computations can be controlled by changing the time series interval size. These independent tasks can then be distributed to multiple processors.

References

[Ain88] William A. Ainsworth. *Speech Recognition by Machine.* Peter Peregrinus Ltd., London, 1988.

[CC87] James Clifford and Albert Croker. The historical relational data model (hrdm) and algebra based on lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, California, 1987. IEEE Computer Society Press.

[CCT94] J. Clifford, A. Croker, and A. Tuzhilin. On completeness of historical relational data models. *ACM Transactions on Database Systems*, 19(1), 1994.

[Cla73] W. B. Clapham, Jr. *Natural Ecosystems.* Macmillan Publishing Company, New York, 1973.

[FPSM91] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. The AAAI Press, Menlo Park, California, 1991.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, New York, 1989.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[Ker86] Fred N. Kerlinger. *Foundations of Behavioral Research.* Holt, Rinehart and Winston, New York, third edition, 1986.

[LC78] Robert E. Larson and John L. Casti. *Principles of Dynamic Programming.* Marcel Dekker Inc., Basel, Switzerland, 1978.

[LR78] Jeffrey B. Little and Lucien Rhodes. *Understanding Wall Street.* Liberty Publishing Company, Cockeysville, Maryland, 1978.

[Odu71] E. P. Odum. *Fundamentals of Ecology.* W. B. Saunders Company, Philadelphia, third edition, 1971.

[Pac90] Norman H. Packard. A genetic learning algorithm for the analysis of complex data. *Complex Systems*, pages 543–572, 1990.

[Poo88] H. Vincent Poor. *An Introduction to Signal Detection and Estimation.* Spinger-Verlag, New York, 1988.

[RL90] Lawrence R. Rabiner and Stephen E. Levinson. Isolated and connected word recognition—theory and selected applications. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 115–153. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[SC90] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 159–165. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[TCG+93] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings Publishing Company, Redwood City, California, 1993.