An Overview of Dynamic Classification: Mining Collections of Trajectories

**Robert Grossman, Magnify, Inc. and University of Illinois at Chicago and
Stuart Bailey, University of Illinois at Chicago**

**Robert Grossman, Magnify, Inc., 100 So. Wacker Dr., Suite 1130, Chicago, Illinois 60606
National Center for Data Mining, University of Illinois at Chicago, M/C 249,
851 S. Morgan Street, Chicago, IL 60607**

KEY WORDS: data mining, dynamic classification, trajectories, time varying data

Abstract

An important challenge in data mining is to extend some of the standard data mining algorithms from data, which is static to data, which is time varying. Phrased slightly different, the challenge is to extend data mining algorithms from data sets consisting of vectors to data sets consisting of time varying paths or trajectories.

We are interested in the problem of classifying time varying data. We consider the special case in which the time varying data, which arises by sampling trajectories from an underlying family of parameterized dynamical systems. We call this the dynamic classification problem. This problem occurs frequently in practice. For example, there are important applications to risk management, robotics, and aeronautics. We describe a new algorithm to attack this problem and provide experimental data from these domains to demonstrate its scalability on collections ranging in size from 1 GB to 40 GBs.

Introduction

A fundamental problem is to mine time varying data. In this paper, we introduce an algorithm for mining time varying data and give experimental data indicating its scalability from small data sets (100 MBytes) to large data sets (40 Gigabytes).

Suppose we have a finite number of classes, say 10,

$$c_1, ..., c_{10},$$

and some data points d_j, say 100,000.

Assume we have a learning set in which each data point is assigned one of the ten classes:

$$(d_1,c_{i1}), ..., (d_{100000}, c_{i100000})$$

A standard problem in data mining is to construct classifiers *automatically,* that is a map f

$$c = f(d)$$

which assigns a class c to each data point d.

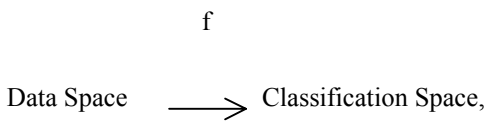
There has been a great deal of previous work on this problem for the case where the data d is vector valued. Our interest here is in the case where the data d is time varying. In particular, we are concerned with the case in which the data arises by sampling a trajectory of a solution of a differential equation. For this we reason, we call this special case *Dynamic Classification*.

Our approach is based upon two key ideas:

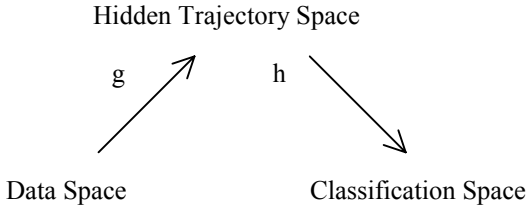
1. We use a hidden space of trajectories and factor the map f through this hidden space. By a trajectory we mean a solution of a differential equation.

2. Our learning algorithm is essentially a nearest neighbor algorithm in the hidden space of trajectories. It differs in two fundamental ways from traditional nearest neighbor algorithms: First, as we just mentioned, it takes place in an auxiliary hidden space rather than the original data space. Second, most nearest neighbor algorithms divide the space isotropically. We use the fact that we are working in a space of trajectories to divide the space non-isotropically in a fashion, which naturally reflects the underlying dynamics of the trajectories.

In other words, rather than directly constructing a classifying map f:



we introduce an auxiliary space of hidden trajectories and factor the map f into the composition of two other maps g and h:



As an important motivating example, consider the problem of identifying fraudulent credit card transactions [Grossman 1996b and 1996c]. One of the most important indicators of possible fraudulent use is the number of transactions per unit time, its derivatives, and its smoothed variants. A common practice is to code this information in an ad hoc fashion and then use it to extract rules, build neural nets, or compute classification trees. As an alternative, one could view this information as representing a parameterized path in (some) phase space. The goal of a classifier would then be to separate the normal trajectories from the fraudulent ones. Since the data is time varying, it is natural to use differential equations to model the dynamics.

We feel that this paper makes a contribution by introducing a new, scalable algorithm for classifying time varying data. In this note we simply provide an overview of this algorithm. For more details, see the companion article [Grossman 1998].

Related Work

There has been recent work in the KDD community on mining time varying data including [Agrawal 1995], [Srikant 1996], and [Mannila 1998]. Our approach is different in that we restrict attention to the special case in which the data arises from an underlying dynamical system and we are interested in algorithms for mining time varying data which exploits this structure.

Transform methods, especially Fourier transforms and their variants, are a traditional means of finding patterns in time series data. Other specialized techniques have been used in the KDD community, including techniques from the speech community [Berndt 1996].

Some of the earliest work in artificial intelligence and knowledge discovery concerned the problem of given experimental data to discover the equations describing that data [Bobrow 1985]. For example, to discover Kepler's laws from planetary data. Our work is related to this work, but differs in two fundamental ways: First, we restrict to parameterized families of differential equations to make the problem more tractable. Second, our approach is data-intensive instead of compute-intensive.

Our approach is based upon a function for hashing trajectories and paths. The function is defined using a sequence of grids in phase space, which divide up phase space non-isotropically in a dyadic fashion, depending upon the properties of the dynamical system. Variants of this algorithm have been previously described in [Grossman 1990, 1992, and 1996a]. This is the first explicit adaptation of this algorithm for classification and directed towards the data mining community.

Breaking up phase space in a dyadic function to reflect the smoothness properties of functions is a old and fundamental idea in analysis. For an exposition and some references, see [Stein 1970, pages 16-20].

Basic Idea

Hashing Trajectories with Dyadic Grids. A *trajectory* through x^0 in R^N is simply a solution of a differential equation

$$d(x(t))/dt = F(x(t), t), \qquad x(0) = x^0 \text{ in } R^N. \qquad (1)$$

For the purposes here, the vector field F defining the differential equation may be thought of simply as a differentiable map

$$F: R^N \longrightarrow R^N$$

defined in a neighborhood of the point x^0 in R^N. As used below, a path is simply a curve in R^N which or may not be a solution of the differential equation.

The basic idea is to use a sequence of nested grids to assign indices in a natural way to solutions of the differential equation [Grossman 1990 and 1992]. A dyadic sequence of grids, or *dyadic grid* for short, is a

nested sequence of grids g_1, g_2, g_3, ..., where the length of each side in grid g_{j+1} is *half* of the length of the corresponding side in grid g_j. For example, in the two dimensional space in Figure 1, the A-grid has four cells, the B-grid has 16 cells, and C-grid has 64 cells and the D-grid has 256 cells. The cells in the grid are numbered consecutively.

A trajectory is broken into sub-trajectories, depending upon the complexity of the dynamics. The more complicated the dynamics, the finer the grid that is used. Each sub-trajectory is assigned an index corresponding to the cell of the appropriate sub-grid. For example, the trajectory in Figure 1 is broken into ten sub-trajectories using the grids A-D and assigned the sequence of indices

C-17, C-9, D-67, D-84, B-6, A-2,

C-39, C-47, C-48, C-40

The indices can then be used for hashing segments of paths and trajectories, so that they can be returned in constant time. Splitting a path or trajectory into segments using dyadic grids can be done in several different ways; for details, see [Grossman 1998].

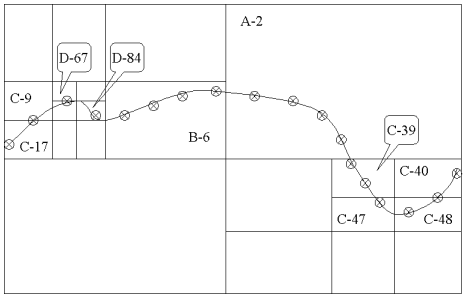


Figure 1. The figure above uses a nested sequence A, B, C, and D of grids to assign indices to a trajectory segment. Loosely speaking, the more rapidly the trajectory changes, as measured by its tangent vector, the finer the grid used.

The next grid in the sequence is obtained by dividing each side of the parent grid cell in half.

Dynamic Classification. The dynamic classification algorithm requires an auxiliary space of hidden trajectory segments. We fill the space of hidden trajectory segments with data from a learning set consisting of trajectory segments, each of which is labeled with a class.

Given a path we proceed as follows:

1. Break up the path into path segments using the dyadic sequence of grids.

2. For each path segment, retrieve all corresponding trajectory segments from the learning set. Select the trajectory segment (and attached class label) which is closest to the path segment.

3. This produces a sequence of trajectory segments, with attached class label, from the space of hidden trajectory segments. Combine the corresponding class labels using majority vote, or other combining algorithm. There are a variety of different combining algorithms; for a survey, see [Dietterich 1997].

Note that for any M 1, we can choose an initial grid with the property that it is fine enough so that each cell in the space of hidden trajectory segments contains fewer than M trajectory segments. In this way, we can associate a class to each path that can be computed in constant time with respect to the size of the collection.

Experimental Studies

Our primary interest was in testing the scalability of hashing paths and trajectories using a sequence of dyadic grids. Once this is achieved, the Dynamic Classification Algorithm sketched above can assign a class to a path or trajectory in constant time, independent of the number of trajectory segments stored in the space of hidden trajectory segments. To test this, we created five collections of trajectory segments, ranging in size from 1 GB to 40 GBs, and retrieved 10 paths and averaged the results. The results (adopted from [Grossman 1996a]) is reported in Table 1.

As a further test, for the 40 GB collection of trajectory segments, we ran tests where we varied the length of the query path, from 20 points to 2000 points. The average time to complete 10 such queries (adopted from [Grossman 1996a]) is reported in Table 2.

Experiments were conducted on a cluster of four workstations connected with Asynchronous Transfer Mode (ATM) technology. Each workstation was a Hewlett-Packard Apollo 9000 Model 725/100 with

128 Megabytes of RAM and a 9 Gigabyte, external, Fast-Wide, single-ended, SCSI disk. The workstations were connected by a FORE Systems ASX-200 ATM switch utilizing Hardware version 1.0 and Software version ForeThought 3.4.2 (1.3). Each workstation was connected to the switch via a Hewlett-Packard J2802B EISA/ATM adapter card rated at OC-3 (155Mbps) and two multi-mode fibers. The communication protocol was HP implemented TCP/IP over ATM (i.e. Classical IP).

Size of Collection of Trajectory Segments (GBs)	**Average Query Time to Return Approximating Sequence (seconds)**
1	2.3
5	2.1
10	2.1
20	2.1
40	2.1

Table 1. Given a query path, the dynamic grid algorithm is designed to retrieve a sequence of approximating trajectory segments with attached class information within a fixed time, independent of the number of trajectory segments in the collection. Experimental results are reported for five different collections of trajectory segments. Results are averaged over ten different paths. The table is adapted from [Grossman 1996a].

Size (GBs)	**Length 20 (seconds)**	**Length 200 (seconds)**	**Length 2000 (seconds)**
1	2.3	17.3	174.2
5	2.1	16.7	169.6
10	2.1	17.0	180.2
20	2.1	16.7	173.2
40	2.1	17.5	171.4

Table 2. A second set of experiments was done to demonstrate that the dynamic grid algorithm scales appropriately as the length of the query path increases. Again, results are averaged for ten different query paths. The table is adapted from [Grossman 1996a].

Summary and Conclusion

Developing classification algorithms for time varying data is a fundamental problem in data mining. We consider the special case in which the time varying data is assumed to arise by sampling trajectories of differential equations. For this special case, we describe a classification algorithm and give some experimental evidence of its scalability.

Our approach is data intensive instead of compute intensive in the sense we store a large collection of trajectory segments labeled with class information and provide an efficient algorithm for retrieving near by trajectory segments. The algorithm hashes the trajectory segments using a dyadic sequence of finer and finer grids.

In this note, we have provided an introduction and overview of dynamic classification. For more details, see the companion article [Grossman 1998].

References

[Agrawal 1995] R. Agrawal and R. Srikant, Mining Sequential Patterns, in Proceedings of the Eleventh International Conference on Data Engineering (ICDE '95), Taipei, Taiwan, pages 3-14, 1995.

[Berndt 1996] D. J. Berndt and J. Clifford, Finding Patterns in Time Series: A Dynamic Programming Approach, in Advances in Knowledge Discovery and Data Mining, edited U. M Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press/MIT Press, pp. 229--248, 1996.

[Bobrow 1985] D. G. Bobrow, editor, Qualitative Reasoning About Physical Systems, MIT Press, Cambridge, 1985.

[Dietterich 1997] Machine Learning Research: Four Current Directions, to appear.

[Grossman 1990] R. Grossman, "Querying databases of trajectories of differential equations I: data structures for trajectories," Proceedings of the 23rdHawaii International Conference on Systems Sciences, IEEE, 1990, pp. 18-23.

[Grossman 1992] R. Grossman, S. Mehta, and X. Qin, Path planning by querying persistent stores of trajectory segments, Laboratory for Advanced Computing Technical Report, Number LAC 93-R3, University of Illinois at Chicago, September, 1992.

[Grossman 1996a] S. Bailey, R. L. Grossman, L. Gu, and D. Hanley, "A Data Intensive Approach to Path Planning and Mode Management for Hybrid Systems," in R. Alur, T. A. Henzigner, and E. Sontag, Hybrid Systems III, Proceedings of the DIMACS Workshop on Verification and Control of Hybrid Systems, Springer-Verlag, LNCS 1066, 1996.

[Grossman 1996b] R. L. Grossman and H. V. Poor, Optimization Driven Data Mining and Credit Scoring, Proceedings of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), IEEE, Piscataway, 1996, pages 104-110.

[Grossman 1996c] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, "Data Mining and Tree-based Optimization," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

[Grossman 1998] R. L. Grossman and S. Bailey, Dynamic Classification: Mining Collections of Trajectory Segments, UIC Center for Data Mining Technical Report TR 98-5, 1998.

[Guckenheimer 1986] J. Guckenheimer and P. Holmes, Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields, Springer-Verlag, New York, 1986.

[Mannila 1997] H. Mannila, H. Toivonen, and I. Verkamo, Discovery of Frequent Episodes in Event Sequences, Data Mining and Knowledge Discovery, Volume 1, pages 259-289, 1997.

[Stein 1970] E. M. Stein, Singular Integrals and Differentiability Properties of Functions, Princeton University Press, 1970.

[Srikant 1996] R. Srikant and R. Agrawal, Mining Sequential Patterns: Generalization and Performance Improvements, in Advances in Database Technology - Fifth International Conference On Extending Database Technology (EDBT '96), Avignon, France, pages 3-17, 1996.