

Discretization of Time Course Data

Elena S. Dimitrova^{*}, John J. McGee, Reinhard C. Laubenbacher

Virginia Bioinformatics Institute at Virginia Tech, Washington St. (0477), Blacksburg, VA 24061, USA

ABSTRACT

Data discretization, also known as binning, is a frequently used technique in computer science, statistics, and their applications to biological data analysis. We present a new method for the discretization of real-valued time courses of data into a finite number of discrete values. It is specifically designed for discrete modeling methods using such data. Novel aspects are the incorporation of an information-theoretic criterion and a criterion to determine the optimal number of values. While the method can be used on other types of data, the motivation for its development is the need for a discretization algorithm for several short multivariate time courses of heterogeneous data, such as transcript, protein, and metabolite concentration measurements. Furthermore, the method needs to preserve the dynamic features of the time courses as well as to be robust to noise in the experimental data. A C++ implementation of the algorithm is available at <http://polymath.vbi.vt.edu/discretization>.

1 INTRODUCTION

Discretization of real data into a typically small number of finite values is often required by machine learning algorithms (Dougherty *et al.*, 1995), data mining (Han, 2000), discrete dynamic Bayesian network applications (van Berlo, 2003), and any modeling algorithm using discrete-state models. Binary discretizations are the simplest way of discretizing data, used, for instance, for the construction of Boolean network models for gene regulatory networks (Kauffman, 1969; Albert and Othmer, 2003). The expression data are discretized into only two qualitative states as either present or absent. An obvious drawback of binary discretization is that labeling the real-valued data according to a present/absent scheme generally causes the loss of a large amount of information. Discrete models and modeling techniques allowing multiple states have been developed and studied in, *e.g.*, (Laubenbacher and Stigler, 2004; Thieffry and Thomas, 1998).

But experimental data are typically continuous, or, at least, represented by computer floating point numbers. For the

case of small samples of biological data, many statistical methods for discretization, such as (Pe'er, 2001), are not applicable due to the insufficient amount of the data. Some other existing discretization techniques assume that the number of discrete classes to be obtained is given, *e.g.*, (Friedman *et al.*, 2000). While this number is extremely important, it is not clear how to properly select it in many cases. Data discretization is a crucial issue for the construction of discrete dynamic models, but has not received the attention it deserves. This paper is a first attempt to address this issue.

We introduce a new method for the discretization of experimental data into a finite number of states. While of interest for other purposes, this method is designed specifically for the discretization of multivariate time courses, such as those used for the construction of discrete models of biochemical networks built from time courses of experimental data. An important characteristic of such time courses that we kept in mind while constructing the discretization method is the relatively small number of data points – typically no more than ten. We employ a graph-theoretic clustering method to perform the discretization and an information-theoretic technique to minimize loss of information content. One of the most useful features of our method is the determination of an optimal number of discrete states that is most appropriate for the data. Our C++ program takes as input one or more vectors of real data and discretizes their entries into a number of states that is most suitable for the data. For the cases when the discretized data are to be fitted by a time-discrete dynamical system, we developed an algorithm which guarantees to eliminate any inconsistencies caused by the discretization process and allows the construction of a deterministic dynamical system containing the discretized data points. Our main objective was to construct a method that is capable of preserving information about network dynamics inherent in the time courses as well as performing well in the presence of noise in the experimental data. We have validated the method in two ways: by comparing the dynamics of a discrete and continuous model constructed using the modeling method in (Laubenbacher and Stigler, 2004) and by using both published and simulated DNA microarray data to test for effectiveness in the presence of noise.

^{*} Corresponding author. Tel.: 540-231-3965; Fax: 540-231-2606

Email addresses: edimitro@vbi.vt.edu (Elena Dimitrova),
jmcgee@vbi.vt.edu (John McGee), reinhard@vbi.vt.edu
(Reinhard Laubenbacher).

2 DISCRETIZATION PROBLEM

In order to place our method in a general context we first give a definition of discretization (Hartemink, 2001):

A *discretization* of a real-valued vector $\mathbf{v} = (v_1, \dots, v_N)$ is an integer-valued vector $\mathbf{d} = (d_1, \dots, d_N)$ with the following properties:

- (1) Each element of \mathbf{d} is in the set $\{0, 1, \dots, D - 1\}$ for some (usually small) positive integer D , called the *degree* of the discretization.
- (2) For all $1 \leq i, j \leq N$, we have $d_i \leq d_j$ if $v_i \leq v_j$.

Spanning discretizations of degree D are a special case that we consider in this paper. They are defined in (Hartemink, 2001) as discretizations that satisfy the additional property that the smallest element of \mathbf{d} is equal to 0 and that the largest element of \mathbf{d} is equal to $D - 1$. We will also assume that for each integer a with $0 \leq a \leq D - 1$, there is an entry d_i of \mathbf{d} such that $a = d_i$. I.e., if sorted, the entries of \mathbf{d} are consecutive integers.

Most discretization methods assume extra knowledge about the data source, which may not always be available. For example, the sample size may be insufficient to estimate distributions. For time courses of transcript data the number of time points is typically much smaller than the number of genes considered, so that statistical approaches to discretization become problematic. Also, in these cases it is rarely known what the appropriate discretization thresholds for each gene might be.

Another common discretization technique is based on *clustering* (Jain and Dubes, 1988). One of the most often used-clustering algorithms is the *k-means* developed by MacQueen (1967). It is a non-hierarchical clustering procedure whose goal is to minimize dissimilarity in the elements within each cluster while maximizing this value between elements in different clusters. Many applications of the *k-means* clustering such as the *MultiExperiment Viewer* (Saeed et al., 2003) start by taking a random partition into k clusters and computing their centroids. As a consequence, a different clustering of S may be obtained every time the algorithm is run. Another inconvenience is that the number k of clusters to be formed has to be specified in advance. Although there are methods for choosing “the best k ” such as the one described in (Crescenzi, 2001), they rely on some knowledge of the data properties that may not be available. Another method is *single-link clustering* (SLC) with the Euclidean distance function. SLC is a divisive (top-down) hierarchical clustering that defines the distance between two clusters as the minimal distance of any two objects belonging to different clusters (Jain and Dubes, 1988). In the context of discretization, these objects will be the real-valued entries of the vector to be discretized, and the distance function that measures the distance between two vector entries v

and w will be the one-dimensional Euclidean distance $|v - w|$. Top-down clustering algorithms start from the entire data set and iteratively split it until either the degree of similarity reaches a certain threshold or every group consists of one object only. For the purpose of data analysis, it is impractical to let the clustering algorithm produce clusters containing only one real value. The iteration at which the algorithm is terminated is crucial since it determines the degree of the discretization, and one of the most important features of our discretization method is a built-in termination criterion.

SLC with the Euclidean distance function satisfies one of our major requirements: very little starting information is needed – only distances between points. In addition, being a hierarchical clustering procedure it lends itself to adjustment in case that clusters need to be split or merged. It may result, however, in a discretization where most of the points are clustered into a single partition if they happen to be relatively close to one another. This negatively affects the information content of the discrete vector (to be discussed later in the paper). Another problem with SLC is that its direct implementation takes D , the desired number of discrete states, as an input. However, we would like to choose D as small as possible, without losing information about the system dynamics and the correlation between the variables, so that an essentially arbitrary choice is unsatisfactory. These two issues were addressed by modifying the SLC algorithm: our method begins by discretizing a vector in the same way as SLC but instead of providing D as part of the input, the algorithm contains termination criteria which determine the appropriate number D . After that each discrete state is checked for information content and if it is determined that this content can be considerably increased by further discretization (to be discussed later), then the state is separated into two states in a way that may not be consistent with SLC. The details of these procedures are given next.

3 METHOD

The method assumes that the data to be discretized consist of one or several vectors of real-valued entries. It is appropriate for applications when there is no knowledge about distribution, range, or discretization thresholds of the data and arranges the data points into clusters only according to their relative distance with respect to each other and the resulting information content. The algorithm employs graph theory as a tool to produce a clustering of the data and provides a termination criterion.

3.1 Discretization of one vector

Even if more than one vector is to be discretized, the algorithm discretizes each vector independently and for some applications this may be sufficient. The example of such a vector to keep in mind is a time course of expression values for a single gene. If the vector contains m distinct entries, a

complete weighted graph on m vertices is constructed, where a vertex represents an entry and an edge weight is the Euclidean distance between its endpoints. The discretization process starts by deleting the edge(s) of highest weight until the graph gets disconnected. If there is more than one edge labeled with the current highest weight, then all of the edges with this weight are deleted. The order in which the edges are removed leads to components, in which the distance between any two vertices is smaller than the distance between any two components, a requirement of SLC. We define the distance between two components G and H to be $\min\{|g-h| \mid g \in G, h \in H\}$. The output of the algorithm is a discretization of the vector, in which each cluster corresponds to a discrete state and the vector entries that belong to one component are discretized into the same state.

3.2 Example

Suppose that vector $\mathbf{v} = (1, 2, 7, 9, 10, 11)$ is to be discretized. The corresponding SLC dendrogram that would be obtained by SLC algorithms such as the Johnson's algorithm (Johnson, 1967) is given in Figure 1.

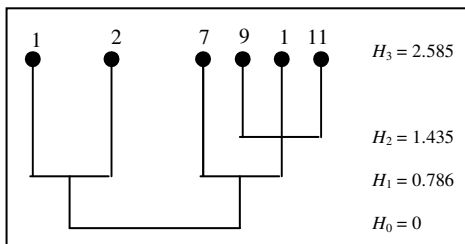


Fig.1. Dendrogram representing the SLC algorithm applied to the data of Example 3.2. The column on the right gives the corresponding Shannon entropy increasing at each consecutive level.

We start by constructing the complete weighted graph based on \mathbf{v} which corresponds to iteration 0 of the dendrogram on Figure 1. Having disconnected the graph, the next task is to determine if the obtained degree of discretization is sufficient; if not, the components need to be further disconnected in a similar manner to obtain a finer discretization. A component is further disconnected if and only if both (1) and (2) hold:

- (1) The minimum vertex degree of the component is less than the number of its vertices minus 1. The contrary implies that the component is a complete graph by itself, *i.e.* the distance between its minimum and maximum vertices is smaller than the distance between the component and any other component.
- (2) One of the following three conditions is satisfied (“disconnect further” criteria):
 - The average edge weight of the component is greater than half the average edge weight of the complete graph.

- The distance between its smallest and largest vertices is greater than or equal to half this distance in the complete graph. For the complete graph, the distance is the graph's highest weight.
- Finally, if the above two conditions fail, a third one is applied: disconnect the component if it leads to a substantial increase in the information content carried by the discretized vector.

The result of applying only the first two criteria is analogous to SLC clustering with the important property that the algorithm chooses the appropriate level to terminate. Applying the third condition, the information measure criterion may, however, result in a clustering which is inconsistent with any iteration of the SLC dendrogram. This criterion is discussed next.

3.3 Information measure criterion

Discretizing the entries of a real-valued vector into a finite number of states certainly reduces the information carried by the discrete vector in the sense defined by Shannon (1948). In his paper, Shannon developed a measure of how much information is produced by a discrete source. The measure is known as *entropy* or *Shannon entropy*. Suppose there is a set of n possible events whose probabilities of occurrence are known to be p_1, p_2, \dots, p_n . Shannon proposed a measure of how much choice is involved in the selection of the event or how certain one can be of the outcome, which is given by

$$H = -\sum_{i=1}^n p_i \log_2 p_i.$$

The base 2 of the logarithm is chosen so that the resulting units may be called bits.

In our context the Shannon entropy of a vector discretized into n states is given by

$$H = \sum_{i=0}^{n-1} \frac{w_i}{n} \log_2 \left(\frac{n}{w_i} \right),$$

where w_i is the number of entries discretized into state i . An increase in the number of states implies an increase in entropy, with an upper bound of $\log_2 n$. However, we want the number of states to be small. That is why it is important to notice that H increases by a different amount depending on which state is split and the size of the resulting new states. For example, if a state containing the most entries is split into two new states of equal size, H will increase more than if a state of fewer entries is split or if we split the larger state into two states of different sizes.

To see that splitting a given state into two states of equal size results in maximum entropy increase, consider a vector

whose entries have been divided into n states, one of which, labeled with 0, contains w_0 entries. As a function of w_0 , the entropy is given by

$$H(w_0) = \frac{w_0}{n} \log_2 \left(\frac{n}{w_0} \right) + \sum_{i=1}^n \frac{w_i}{n} \log_2 \left(\frac{n}{w_i} \right).$$

Suppose that we split state 0 into two states containing m and $w_0 - m$ entries, respectively, where $0 < m < w_0$. This will change only the first term of the right-hand side of the above entropy expression and leave the summation part the same. It is easy to verify that $h(w_0) = \frac{w_0}{n} \log_2 \left(\frac{n}{w_0} \right)$

achieves its maximum value over $0 < m < w_0$ at $m = \frac{w_0}{2}$.

Therefore, splitting a state into two states of equal size maximizes the entropy increase.

As explained in the previous section, the information measure criterion is applied to a component only after the component has failed the other three conditions. Once this happens, we consider splitting it further only if doing so would provide a very significant increase of the entropy, *i.e.* if the component corresponds to a “large” collection of entries (recurring entries are included since all entries have to be considered when computing the information content of a vector). In our implementation *a component gets disconnected further only if it contains at least half the vector entries*. Unlike with the other criteria, if a component is to be discretized under the information condition, the corresponding sorted entries are split into two parts: not between the two most distant entries but into two equal parts (or with a difference of one entry in case of an odd number of entries). This is to guarantee a maximum increase of the information measure.

In Example 3.2, the two components that were obtained by removing the edges of heaviest weight both fail the “disconnect further” conditions. If the discretization process stopped at this iteration, vector $\mathbf{d} = (0, 0, 1, 1, 1, 1)$ would have Shannon entropy 0.78631. Having most of the entries of \mathbf{v} discretized into the same state, 1, reduces the information content of \mathbf{d} .

Suppose discretization of \mathbf{v} continues according to SLC, *i.e.*, without enforcing the fourth condition of “disconnect further”. The next step is to remove the edges of highest weight until a component gets disconnected. This yields the removal of the four edges of weights 4, 3, 2, and 2, respectively, to obtain $\mathbf{d} = (0, 0, 1, 2, 2, 2)$. The Shannon entropy of the new discretization of \mathbf{v} is 1.43534. Still half of the entries of \mathbf{v} remain at the same discrete level, now 2, which does not allow for a maximal increase in the information content of \mathbf{d} . If instead discretization proceeded by applying the information criterion to the bigger component, the re-

sulting discretization becomes $\mathbf{d} = (0, 0, 1, 1, 2, 2)$ with Shannon entropy 1.58631, as opposed to the previous entropy of 1.43534.

As illustrated by Example 3.2, the proposed discretization algorithm produces a discretization which is consistent with the definition given above, keeps the number of discrete states small, and maximizes information content over traditional SLC.

3.4 Algorithm summary

Input: set $S_r = \{\mathbf{v}_i \mid i = 1, \dots, m\}$ where each $\mathbf{v}_i = (v_{i1}, \dots, v_{iN})$ is a real-valued vector of length N to be discretized.

Output: set $S_d = \{\mathbf{d}_i \mid i = 1, \dots, m\}$ where each $\mathbf{d}_i = (d_{i1}, \dots, d_{iN})$ is the discretization of \mathbf{v}_i for all $i = 1, \dots, m$.

- (1) For each $i = 1, \dots, m$, construct a complete weighted graph G_i where each vertex represents a distinct v_{ij} and the weight of each edge is the Euclidean distance between the incident vertices.
- (2) Remove the edge(s) of highest weight.
- (3) If G_i is disconnected into components $C_{i1}^{G_i}, \dots, C_{iM_i}^{G_i}$, go to 4. Else, go to 2.
- (4) For each $C_{ik}^{G_i}$, $k = 1, \dots, M_i$, apply “disconnect further” criteria 1–3. If any of the three criteria holds, set $G_i = C_{ik}^{G_i}$ and go to 2. Else, go to 5.
- (5) Apply “disconnect further” 4. If criterion 4 is satisfied, go to 6. Else, go to 7.
- (6) Sort the vertex values of $C_{ik}^{G_i}$ and split them into two sets: if $|V(C_{ik}^{G_i})|$ is even, split the first $|V(C_{ik}^{G_i})|/2$ sorted vertex values of $C_{ik}^{G_i}$ into one set and the rest – into another. If $|V(C_{ik}^{G_i})|$ is odd, split the first $|V(C_{ik}^{G_i})|/2 + 1$ sorted vertex values of $C_{ik}^{G_i}$ into one set and the rest – into another.
- (7) Sort the components $C_{ik}^{G_i}$, $k = 1, \dots, M_i$, by the smallest vertex value in each $C_{ik}^{G_i}$ and enumerate them $0, \dots, D_i - 1$, where D_i is the number of components into which G_i got disconnected. For each $j = 1, \dots, N$, d_{ij} is equal to the label of the component in which v_{ij} is a vertex.

3.5 Algorithm complexity

Given M variables, with N time points each, we compute $N(N-1)/2$ distances to construct the distance matrix so the complexity of this step is $O(N^2)$. The distance matrix is used to create the edge and vertex sets of the complete distance graph, containing $N(N-1)/2$ edges. This can also be accomplished in $O(N^2)$ time. These edges are then sorted in decreasing order, so that the largest edges are removed first. A standard sorting algorithm, such as merge sort, has complexity $O(N \log N)$ (Knuth, 1998). As each edge is removed, the check for graph disconnection involves testing for the existence of a path between the two vertices of the edge. This test for graph disconnection can be accomplished with

a breadth-first search, which has order $O(E+V)$ (Pemmaraju, 2003), with E the number of edges and V the number of vertices in the component. In our case this translates to complexity $O(N^2)$. Edge removal is typically performed for a large percentage of the $N(N-1)/2$ edges, so this step has overall complexity $O(N^4)$. The edge removal step dominates the complexity so that the overall complexity is $O(MN^4)$ to discretize all M variables.

While this is the theoretical worst-case performance, because of the heuristics we have added the typical performance is significantly better.

3.6 Requirements on the number of states

Some applications may require that all vectors in a data set be discretized into the same number of states. For example, the approach adopted by Laubenbacher and Stigler (2004) imposes such a requirement on the discretization. The way we deal with this is by first discretizing all vectors separately. Suppose that for N vectors, the method discretized each into m_1, m_2, \dots, m_N states, respectively. Let $m = \max\{m_i \mid i = 1, \dots, N\}$.

Suppose that a vector has been discretized into m_i states, $m_i < m$. Since the discretization algorithm yielded m_i clusters, the remaining $m - m_i$ can be constructed by sorting the entries in each cluster and splitting the one that contains the two most distant entries with respect to Euclidean distance. The splitting should take place between these entries. This is repeated until m clusters are obtained.

This approach has a potential problem. For instance, if a vector got discretized into 4 states and the total number of distinct entries of the vector is 5, then $m = 6$ cannot be reached. In this case m should be set to 5 by merging states when necessary. In general it may not be desirable to reduce the number of states because this results in loss of information. We would rather increase the number of states unless it is impossible as in the above example.

4 PRESERVATION OF DYNAMICS

As mentioned in the introduction, the discretization algorithm is designed to preserve the dynamic features of time course data. Due to limited knowledge of the dynamic features of real biochemical networks, the validation of our method is best done with a simulated network. In order to extract information about the network topology and dynamics from the discretized data we use the network reconstruction method in (Laubenbacher and Stigler, 2004). However, any other network reconstruction method using discrete data could be applied, e.g., dynamic Bayesian network methods. The goal of this validation is to show that the discretized data contain information about the network that was used to generate the continuous data. We used the A-Biochem software system developed by P. Mendes and his collaborators (Mendes *et al.*, 2003) to generate the simulated network. A-Biochem automatically generates artificial gene

networks with particular topological and kinetic properties. These networks are embodied in kinetic models, which are used by the biochemical-network simulator *Gepasi* (Mendes, 1993, 1997) to produce simulated gene expression data. We generated an artificial gene network with five genes and ten total input connections using the Albert-Barabási algorithm (Albert and Barabási, 2000). In Figure 2 the arrows represent gene activation in the arrow's direction and the segments stand for inhibition.

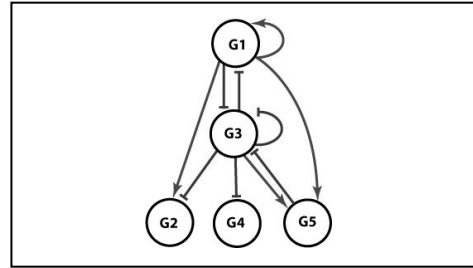


Fig.2. Wiring diagram of the A-Biochem-generated artificial gene network on five genes G_i .

Gepasi uses a continuous representation of biochemical reactions, based on ordinary differential equations (ODE). With the parameters we specified, *Gepasi* generated an ODE system that represents the network. For example, the synthesis rate of gene G_1 is given by

$$\frac{dG_1}{dt} = \frac{0.01 \left(\frac{G_1(t)}{0.01 + G_1(t)} \right)}{0.01 + G_3(t)} - G_1(t).$$

Analyzing the dynamics of the ODE system, one finds that it has two stable steady states (of which only the first is biochemically meaningful): $S_1 = (1.99006, 1.99006, 0.000024814, 0.997525, 1.99994)$ and $S_2 = (-0.00493694, -0.00493694, -0.0604538, -0.198201, 0.0547545)$.

As Laubenbacher and Stigler demonstrated, the performance of their algorithm dramatically improves if *knockout* time courses for genes are incorporated. For this reason we supplied seven time courses of 11 points each: two wild-type time courses and five knockout time courses, one for each gene. The first wild-type time course is generated by solving the ODE system numerically for $t = 0, 2, 6, \dots, 20$ with initial conditions $G_i(0) = 1$ for all $i = 1, \dots, 5$. The top part of Figure 4 shows a plot of the numerical solution of the ODE system with these initial conditions. One can simulate a gene knockout in *Gepasi* by setting the corresponding variable and initial condition to zero. The second time course is generated like the first one but this time with $t = 0, 1, \dots, 10$ and initial conditions $(G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (1, -1, -0.6, -1, 0.5)$. The time points from each of the seven time courses constitute the input vectors. The discretization algorithm chose a state set X of cardinality 5 and, based on the discrete data, the reverse engineering method generated

five polynomials describing the discrete model. For example, the polynomial that describes the dynamics of G_1 (which is now denoted by x_1) is

$$f_1 = -x_1 * x_5^2 - 2 * x_2 * x_5^2 + 2 * x_4 * x_5^2 + 2 * x_5^3 + x_1 * x_2 + x_2^2 - 2 * x_1 * x_3 + x_3^2 - 2 * x_1 * x_4 + 2 * x_2 * x_4 - x_3 * x_4 + x_4^2 - x_2 * x_5 + x_4 * x_5 - x_1 - 2 * x_2 - x_3 - 2 * x_5 - 1.$$

That is, the discrete model is given by a time-discrete dynamical system

$$f = (f_1, \dots, f_5) : X^5 \rightarrow X^5.$$

Now we compare the dynamics of the two models. First, the discretization maps steady state S_1 to the fixed point $FP_1 = (4, 4, 1, 4, 2)$ of f and steady state S_2 to the fixed point $FP_2 = (0, 1, 1, 1, 0)$. The time course produced by solving the ODE system and converging to S_1 is given in the top part of Figure 4. The corresponding discrete points from the time course in the bottom part of Figure 4 form a trajectory that ends at FP_1 (Figure 5). The discrete model trajectory can be superimposed over the discretization of the continuous one, illustrating the matching dynamics of the two models. The same can be observed for the second steady-state S_2 that is mapped to fixed point FP_2 .

Further evidence of the ability of our method to retain information in the data is the fact that the reverse-engineering method in (Laubenbacher and Stigler, 2004) can extract most of the information about the wiring diagram of the network. This can be seen by comparing the inferred diagram in Figure 3 with the diagram of actual direct interactions in Figure 2.

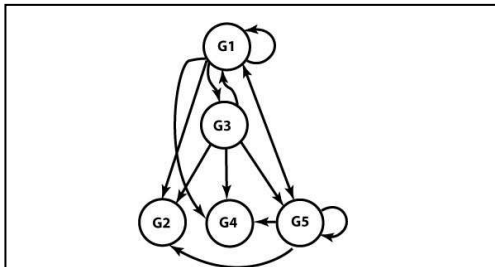


Fig.3. Wiring diagram for reverse engineered system.

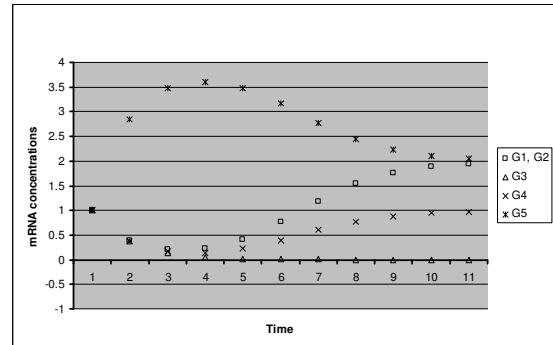
5 DISCRETIZATION IN THE PRESENCE OF NOISE

In the previous example no noise was added to the time courses generated from the artificial gene network. Noise, however, is naturally present in biological data and in microarray data in particular (Hassibi, 2005). Although there are various techniques which increase the accuracy of the microarray measurements, the data inevitably contain errors due to the probabilistic characteristics of the detection process, from sample extraction and mRNA purification to hybridization and imaging. Consequently, it is crucial to study

how the proposed discretization algorithm performs in the presence of typical levels of noise in the experimental data.

5.1 Noiseless data

To study the effect of noise, we considered two types of data. We used gene expression measurements generated for a study of rat cervical central nervous system development (Wen et al., 1998). The data consist of nine expression measurements for each gene: cervical spinal cord tissue was dissected from animals in embryonic development at days 11, 13, 15, 18, and 21 and at postnatal days 0, 7, 14, and 90.



Time	G_1	G_2	G_3	G_4	G_5
0	3	3	4	4	1
2	1	1	3	2	4
4	1	1	2	2	4
6	1	2	1	2	4
8	3	3	1	3	3
10	4	4	1	4	2
12	4	4	1	4	2
14	4	4	1	4	2
16	4	4	1	4	2
18	4	4	1	4	2
20	4	4	1	4	2

Fig. 4. Top: wild-type time course generated by solving numerically the ODE system for $t = 0, \dots, 10$ with initial conditions $(G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (1, 1, 1, 1, 1)$; bottom: corresponding discrete point time course.

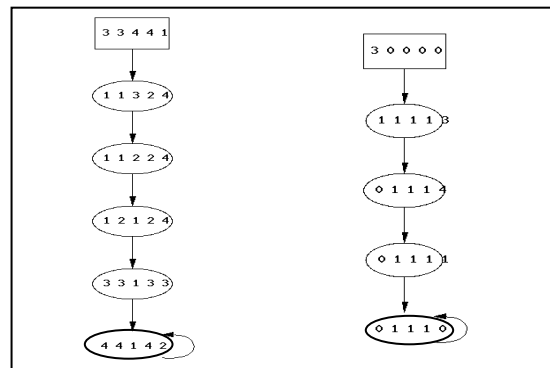


Fig.5. Trajectories formed by the discretized wild-type time courses (Vastani et al., 2004).

We selected 18 genes whose nine-point time courses have statistical variance of more than 1. Although the data likely

contain some noise, here we assume that the measurements are perfect and we add noise of known proportion and distribution assuming it is the only noise in the data. The purpose is to work with data which have the properties of real microarray time courses. The second data set is an artificial ten-point time course on 30 “genes.” The values are randomly generated real numbers in the range [0, 20] with statistical variance greater than 10.

5.2 Adding noise to the data

Two types of noise are added to the data at the same time: *overall* and *point-specific* (de la Fuente, 2004). The overall noise is added by sampling from a normal distribution with zero mean and a standard deviation equal to 12.5% of the standard deviation of each gene. The point-specific noise is simulated by adding noise to each time point sampling from a different normal distribution with zero mean and standard deviation equal to 12.5% of the standard deviation of the particular time point value. Thus, the total proportion of added noise amounts to 25%. For each gene 100 noise containing replicates were generated and discretized.

5.3 Results

For each gene, we compare the way the original noiseless time course was discretized to the discretization of the noise-containing replicates. We count the number of noise-containing replicates that got discretized exactly the same as the original noiseless time course. For the gene expression data from (Wen *et al.*, 1998), 78.72% of all the 18×100 noise-containing replicates were discretized exactly as their corresponding noiseless original. This number is significantly higher for the randomly generated data of variance 10 or higher: 93.8% of all the 30×100 noise-containing replicates were discretized exactly as their corresponding noiseless original. The big difference between the two results can be explained by the different statistical variance of the original data in the two cases. Only 5 out of the 18 genes in the expression measurements time course in (Wen *et al.*, 1998) have variance greater than 10 while the “genes” in the simulated time course had variance of at least 10 with average variance of 34.5. Since the higher the variance of a time course, the better the chance of more distinct and easier to detect discrete states, the discretization of the data with higher variance not surprisingly showed more robustness in the presence of noise.

6 INCONSISTENCIES IN THE DISCRETIZED DATA

One of our objectives is to introduce a discretization method suitable specifically for time courses of data. The algorithm is more general and can be applied to any collection of data points disregarding their order. If, however, the time courses are to be fitted by a deterministic dynamical system, as is the case with most reverse engineering methods, the order

of the points becomes important. If any of the input time courses contain consecutive points $(d_1, \dots, d_N) \rightarrow (a_1, \dots, a_N)$ and $(d_1, \dots, d_N) \rightarrow (b_1, \dots, b_N)$, then we should require that $a_j = b_j$ for all $j = 1, \dots, N$. Even if the experimental data satisfy this requirement, the discretization may create inconsistencies. To obtain a useable time course while at the same time discretizing consistently, several approaches are possible. One is to coarsen the discretization by merging discrete states. Although this would certainly solve the inconsistency problem, it would also reduce the information content of the discrete data. The approach that we adopt is to split as many times as necessary the discrete states of the problematic point (d_1, \dots, d_N) . The state to split first is naturally the one containing the two most distant consecutive real values. Assuming that the analog time course data are consistent, discretization-induced inconsistencies are handled in the following way:

In the discretized time course, find all points (d_1, \dots, d_N) for which there are at least two distinct points (a_1, \dots, a_N) and (b_1, \dots, b_N) that immediately follow point (d_1, \dots, d_N) anywhere in the time course. For each (d_1, \dots, d_N) :

- (1) For each d_j find all real values $x_{j,i}$ that were discretized into state d_j , sort them, and re-index them.
- (2) Let $d_j^{\max} = \max \{|x_{j,i+1} - x_{j,i}|\}$ and let $(x_{j,\text{left}}, x_{j,\text{right}}) = (x_{j,i+1} - x_j)$ for which $|x_{j,i+1} - x_{j,i}| = d_j^{\max}$.
- (3) Let $D_{\max} = \{d_j^{\max} \mid j = 1, \dots, N\}$. Split state d_j that contains values $x_{j,\text{left}}$ and $x_{j,\text{right}}$ for which $x_{j,\text{right}} - x_{j,\text{left}} = D_j^{\max}$. Re-label states accordingly.

Repeat until no inconsistencies are present.

7 DISCUSSION

The method presented here has several novel features. It uses Shannon’s information criterion to determine clusters, identifies the optimal number of clusters for a given data set, and eliminates inconsistencies in the discrete time courses. It is particularly suitable for the discretization of multivariate time courses, since it preserves a large degree of information about dynamic features and ensures data consistency. It thus provides a valuable tool for any application that requires discretization of continuous data when the number of discrete classes that best fits the data is unknown. An important advantage of using discrete states is that a significant portion of the noise is absorbed in the process. The experiments we carried out make us confident that for data that discretize into a relatively small number of states and that contain a degree of noise common to many biological data, the majority of the noise is absorbed into the discrete states.

We do not, however, recommend using the method whenever a large amount of data is available. In this case, a statistical method, such as (Pe’er, 2001), that can take advantage of the statistical properties of the data may be more appropriate. Our method assumes no knowledge of these proper-

ties and therefore cannot utilize this information. Another situation when virtually any discretization technique may not produce useful results is in the case when one or several of the system variables change much more rapidly than the rest. In light of the discussion in Section 6 it is clear that in order to avoid inconsistencies in the discretized data, one may need a very large number of discrete states which may be a disadvantage for the modeling process.

ACKNOWLEDGEMENTS

This work has been partially supported by NIH Grant Nr. RO1GM068947-01. The authors thank A. Jarrah, A. de la Fuente, P. Mendes, and B. Stigler for contributing insights and help with editing, and D. Camacho, I. Hoeschele, and W. Sha for helpful discussions.

REFERENCES

- Albert, R. and Barabási, A. (2000) Topology of evolving networks: local events and universality. *Phys. Rev. Lett.* **85**, 5234–5237.
- Albert, R. and Othmer, H. (2003) The topology of the regulatory interactions predict the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *J. Theor. Biol.* **223**, 1–18.
- Crescenzi, M., Giuliani, A. (2001) The main biological determinants of tumor line taxonomy elucidated by of principal component analysis of microarray data. *FEBS Letters* **507**, 114–118.
- de la Fuente, A., Bing, N., Hoeschele, I., Mendes, P. (2004) Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics* **20**(18), 3565–3574.
- Dougherty, J., Kohavi, R., Sahami, M. (1995) Supervised and unsupervised discrimination of continuous Features. In Frieditis, A. and Russell, S. (eds.), *Machine learning: Proceedings of the 12th International Conference*, Morgan Kaufman, San Francisco, CA.
- Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000) Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **7**, 601–620.
- Han, J., Kamber, M. (2000) *Data Mining: Concepts and Techniques*. Academic Press, San Diego, CA.
- Hartemink, A. (2001) Principled computational methods for the validation and discovery of genetic regulatory networks. Massachusetts Institute of Technology, Ph. D. dissertation.
- Hassibi, A., Vikalo, H. (2005) Probabilistic Modeling and Estimation of Gene Expression Levels in Microarray. *Proc. IEEE Workshop on Genomic Signal Processing and Statistics (GENSIPS)*.
- Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice Hall. 58–89.
- Johnson, S. C. (1967) Hierarchical clustering schemes. *Psychometrika* **32**, 241–254.
- Kauffman, S. A. (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* **22**, 437–467.
- Knuth, D.E., 1998. *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd edition. Addison-Wesley, Reading, Massachusetts.
- Laubenbacher, R. and Stigler, B. (2004) A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol.* **229**, 523–537.
- MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium of Mathematical Statistics and Probability* **1**, 281–297. University of California Press, Berkeley, CA.
- Mendes, P. (1993) GEPASI: a software package for modeling the dynamics, steady states and control of biochemical and other systems. *Comput. Appl. Biosci.* **9**, 563–571.
- Mendes, P. (1997) Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. *Trends Biochem. Sci.* **22**, 361–363.
- Mendes, P., Sha, W., and Ye, K. (2003) Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics* **19**, ii122–ii129.
- Pe'er, D., Regev, A., Elidan, G., Friedman, N. (2001) Inferring subnetworks from perturbed expression profiles. *Bioinformatics* **17**, S215–224.
- Pemmaraju, S., Skiena, S. (2003) *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press.
- Saeed, A., Sharov, V., White, J., Li, J., Liang, W., Bhagabati, N., Braisted, J., Klapa, M., Currier, T., Thiagarajan, M., Sturn, A., Snuffin M., Rezantsev, A., Popov, D., Ryltsov, A., Kostukovich, E., Borisovsky, I., Liu, Z., Vinsavich, A., Trush, V., Quackenbush, J. (2003) TM4: a free, open-source system for microarray data management and analysis. *BioTechniques* **34**(2), 374–378.
- Shannon, C. (1948) A Mathematical Theory of Communication. *The Bell Systems Technical Journal* **27**, 379–423, 623–656.
- Thieffry, D. and Thomas, R. (1998) Qualitative analysis of gene networks. *Proc. Pacific Symp. on Biocomputing*, 77–88, World Scientific, Singapore.
- van Berlo, R., van Someren, E., Reinders, M. (2003) Studying the Conditions for Learning Dynamic Bayesian Networks to Discover Genetic Regulatory Networks. *SIMULATION* **79**(12), 689–702.
- Vastani, H., Jarrah, A., Laubenbacher, R. (2004) Available at <http://dvd.vbi.vt.edu>.
- Wen, X., Fuhrman, S., Michaelis, G., Carr, D., Smith, S., Barker, J., Somogyi, R. (1998) Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci. USA.* **95**, 334–339.