Continuous Representations of Time Series Gene Expression Data

Ziv Bar-Joseph Georg Gerber David K. Gifford
MIT Laboratory for Computer Science
200 Technology Square, Cambridge, MA 02139
{zivbj,georg,gifford}@mit.edu

Tommi S. Jaakkola
MIT Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02139
tommi@ai.mit.edu

Itamar Simon
Whitehead Institute for Biomedical Research
9 Cambridge Center, Cambridge, MA 02142
simon@wi.mit.edu

Abstract

We present algorithms for time-series gene expression analysis that permit the principled estimation of unobserved time-points, clustering, and dataset alignment. Each expression profile is modeled as a cubic spline (piecewise polynomial) that is estimated from the observed data and every time point influences the overall smooth expression curve. We constrain the spline coefficients of genes in the same class to have similar expression patterns, while also allowing for gene specific parameters. We show that unobserved time-points can be reconstructed using our method with 10-15% less error when compared to previous best methods. Our clustering algorithm operates directly on the continuous representations of gene expression profiles, and we demonstrate that this is particularly effective when applied to non-uniformly sampled data. Our continuous alignment algorithm also avoids difficulties encountered by discrete approaches. In particular, our method allows for control of the number of degrees of freedom of the warp through the specification of parameterized functions, which helps to avoid overfitting. We demonstrate that our algorithm produces stable low-error alignments on real expression data and further show a specific application to yeast knockout data that produces biologically meaningful results.

Keywords: Time series expression data, Missing value estimation, Clustering, Alignment.

1 Introduction

Principled methods for estimating unobserved time-points, clustering, and aligning microarray gene expression time-series are needed to make such data useful for detailed analysis. Datasets measuring temporal behavior of thousands of genes offer rich opportunities for computational biologists. For example, Dynamic Bayesian Networks may be used to build models and try to understand how genetic responses unfold. However, such modeling frameworks need a sufficient quantity of data in the appropriate format. Current gene expression time-series data often do not meet these requirements, since they may be missing data points, sampled non-uniformly, and measure biological processes that exhibit temporal variation.

In many applications, researchers may face the problem of reconstructing unobserved gene expression values. Values may not have been observed for two reasons. First, errors may occur in the experimental process that lead to corruption or absence of some expression measurements. Second, we may want to estimate expression values at time points different from those originally sampled. In either case, the nature of microarray data makes straightforward interpolation difficult. Data are often very noisy and there are few replicates. Thus, simple techniques such as interpolation of individual genes can lead to poor estimates. Additionally, in many cases there are a large number of missing time-points in a series for any given gene, making gene specific interpolation infeasible. In the case of clustering, the treatment of time-series can be problematic, as a time-series represents a set of dependent experiments. A particular problem arises when series are not sampled uniformly such as in [16, 4, 7].

Variability in the *timing* of biological processes further complicates gene expression time-series analysis. The rate at which similar underlying processes such as the cell-cycle unfold can be expected to differ across organisms, genetic variants, and environmental conditions. For instance, Spellman $et\ al$ [16] analyze time-series data for the yeast cell-cycle in which different methods were used to synchronize the cells. It is clear that the cycle lengths across the different experiments vary considerably, and that the series begin and end at different phases of the cell-cycle. Thus, one needs a method to align such series to make them comparable.

In this paper we use statistical spline estimation to represent time-series gene expression profiles as continuous curves. Our method takes into account the actual duration each time point represents, unlike most previous approaches that treat expression time-series like static data consisting of vectors of discrete samples [7, 12, 9]. Our algorithm generates a set of continuous curves that can be used directly for estimating unobserved data. However, although our method uses spline curves (piecewise polynomials) to represent gene expression profiles, it is not reasonable to fit each gene with an individual spline due to the issues with microarray datasets discussed above. Instead, we constrain the spline coefficients of genes in the same class to covary similarly, while also allowing for gene specific parameters. A class is a set of genes with similar expression profiles that may be constructed using prior biological knowledge or clustering methods. We present a clustering algorithm that infers classes automatically by operating directly on the continuous representations of expression profiles. This is particularly effective when applied to non-uniformly sampled data. However, note that our method does require data that has been sampled at a sufficiently high rate. We demonstrate in Section 6 that our method performs well on several such datasets, but for other datasets that have been sampled at rates too low to capture changes in the underlying biological processes, our method will not be effective. A future direction would be to use our method to determine the quality of the sampling rate.

Our alignment algorithm uses the same spline representation of expression profiles to continuously time-warp series. First, a parameterized function is chosen that maps the time-scale of one series into another.

Because we use parameterized functions, we are explicitly specifying the number of allowed degrees of freedom, which is helpful in avoiding overfitting. Our algorithm seeks to maximize the similarity between the two sets of expression profiles by adjusting the parameters of the warping function.

The remainder of this paper is organized as follows. In Section 2 we present a brief introduction to splines. In Section 3 we discuss our algorithm for estimating unobserved data and in Section 4 we extend this algorithm to perform clustering. In Section 5 we present our alignment algorithm. Section 6 presents applications of our method to expression data and Section 7 concludes the paper and suggests directions for future work.

1.1 Related Work

Recently, several papers have focused on modeling and analyzing the temporal aspects of gene expression data. In Holter $et\ al$ [10] a time translational matrix is used to model the temporal relationships between different modes of the Singular Value Decomposition (SVD). Unlike our work, this method focuses on the SVD modes and not on specific genes. In addition, only relationships between time points that are sampled at the lowest common frequencies can be studied. Thus, not all available expression data can be used. In Zhao $et\ al$ [20] a statistical model is fit to all genes in order to find those that are cell cycle regulated. This method uses a custom tailored model, relying on the periodicity of the specific dataset analyzed, and is thus less general than our approach.

Several papers have used simple interpolation techniques to estimate missing values for gene expression data. Aach $et\ al$ [1] use linear interpolation to estimate gene expression levels for unobserved time-points. D'haeseleer [6] use spline interpolation on individual genes to interpolate missing time-points. As we show in Section 6.2, both techniques cannot approximate the expression curve of a gene well, especially if there are many missing values. In Troyanskaya $et\ al$ [17] several techniques for missing value estimations were explored. However, none of the suggested techniques take into account the actual times the points correspond to, and thus time series data is treated in the same way as static data. As a consequence, their techniques cannot estimate values for time-points between those measured in the original experiments.

There is a considerable statistical literature that deals with the problem of analyzing non-uniformly sampled data. These models, known as mixed-effect models [3] use spline estimation methods to construct a common class profile for their input data. Recently, James and Hastie [11] presented a reduced rank mixed effects model that was used for classifying medical time-series data. In this paper we extend these methods to gene expression data. Unlike the above papers, we focus on the gene specific aspects rather than the common class profile. In addition, we present a method that is able to deal with cases in which class membership is not given. Another difference between this work and [11] is that we do not use a reduced rank approach, since gene expression datasets contain information about thousands of genes.

Many clustering algorithms have been suggested for gene expression analysis (see [13]). However, as far as we are aware, all these algorithms treat their input as a vector of data points, and do not take into account the actual times at which these points were sampled. In contrast, our algorithm weights time points differently according to the sampling rate.

Aach $et\ al$ [1] presented a method for aligning gene expression time-series that is based on Dynamic Time Warping, a discrete method that uses dynamic programming and is conceptually similar to sequence alignment algorithms. Unlike with our method, the allowed degrees of freedom of the warp operation in Aach $et\ al$ depends on the number of data points in the time-series. Their algorithm also allow mappings of multiple time-points to a single point, thus stopping time in one of the datasets. In contrast, our algorithm

avoids temporal discontinuities by using a continuous warping representation. There is also a substantial body of work in the speech recognition and computer vision community that deals with data alignment. For instance, non-stationary Hidden Markov models with warping parameters have been used for alignment of speech data [5], and mutual information based methods have been used for registering medical images [18]. However, these methods generally assume high resolution data, which is not the case with available gene expression datasets.

2 Splines

Splines are piecewise polynomials with boundary continuity and smoothness constraints. They are widely used in fields such as computer-aided design, image processing and statistics [2, 19, 8, 11]. The use of piecewise low-degree polynomials results in smooth curves and avoids the problems of overfitting, numerical instability and oscillations that arise if single high-degree polynomials were used. In this paper we use cubic splines since they have a number of desirable properties. For instance, cubic polynomials are the lowest degree polynomials that allow for a point of inflection. Thus, we will restrict the subsequent discussion to the cubic case.

A cubic spline can be represented with the following equation:

$$y(t) = \sum_{i=1}^{n} C_i S_i(t) \qquad t_{min} \leq t < t_{max} \tag{1}$$

Here, t is the parameter (e.g., time), $S_i(t)$ are polynomials, and C_i are the coefficients. The typical way to represent a piecewise cubic curve is simply:

$$S_{4j+l}(t) = \begin{cases} t^{l-1}, & x_j \leq t \leq x_{j+1} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Here, $l = 1 \ldots 4$, $j = 0 \ldots n/4 - 1$ and the x_j's denote the *break-points* of the piecewise polynomials. Thus, we have $n/4$ cubic polynomials that can be denoted $p_j(t) = \sum_{l=1}^{4} C_{4j+l} t^{l-1}$. In order to determine the coefficients of these polynomials, n equations are required. If one specifies a value D_j plus continuity constraints up to the second derivative for the piecewise polynomial at each break-point x_j for $j = 1 \ldots n/4 - 1$, four equations are obtained for each of the $n/4 - 1$ internal break-points:

$$\begin{aligned} p_j(x_j) &= D_j \\ p_j(x_j) &= p_{j-1}(x_j) \\ p'_j(x_j) &= p'_{j-1}(x_j) \\ p''_j(x_j) &= p''_{j-1}(x_j) \end{aligned}$$

Additionally, specifying values for the end-points $p_0(x_0) = D_0$ and $p_{n/4-1}(x_{n/4}) = D_{n/4}$ yields a total of $n - 2$ equations. Thus, in order to solve for the spline coefficients, an additional two equations are needed. Typically, these equations are obtained by specifying the first or second derivatives at the two end-points of the spline. Note that since one explicitly specifies the values $p_j(x_j)$ at the break-points, the formulation in equation 2 is particularly useful for defining *interpolating* splines.

2.1 B-splines

While the method discussed so far for defining cubic splines is easy to understand, it is not the most flexible or mathematically convenient formulation for many applications. Alternately, one can write a cubic polynomial in terms of a set of four normalized *basis* functions. A very popular basis is the B-spline basis, which has a number of desirable properties. The texts by Rogers and Adams [14] and Bartels *et al* [2] give a full treatment of this topic. Once again, the discussion here will be limited to features relevant to this papers. Most significantly for the application of fitting curves to gene expression time-series data, it is quite convenient with the B-spline basis to obtain approximating or *smoothing* splines rather than interpolating splines. Smoothing splines use fewer basis coefficients than there are observed data points, which is helpful in avoiding overfitting. In this regard, the basis coefficients C_i can be interpreted geometrically as *control points*, or the vertices of a polygon that control the shape of the spline but are not interpolated by the curve. It can be shown that the curve lies entirely within the convex hull of this controlling polygon. Further, each vertex exerts only a local influence on the curve, and by varying the vector of control points and another vector of knot points (discussed below), one can easily change continuity and other properties of the curve.

The normalized B-spline basis can be calculated using the Cox-deBoor recursion formula [14]:

$$b_{i,1}(t) = \begin{cases} 1, & x_i \leq t < x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$b_{i,k}(t) = \frac{(t - x_i)b_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)b_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{4}$$

Here, k is the order of the basis polynomials (i.e. for a cubic polynomial $k = 4$).

The values x_i are called *knots*, where $i = 1 \ldots n + k$. A *uniform* knot vector is one in which the entries are evenly spaced, i.e., $\mathbf{x} = (0, 1, 2, 3, 4, 5, 6, 7)^T$. If a uniform knot vector is used, the resulting B-spline is called periodic since the basis functions will be translates of each other, i.e., $b_{i,k}(t) = b_{i-1,k}(t - 1) = b_{i+1,k}(t + 1)$. See figure 1 for an example. For a periodic cubic B-spline ($k = 4$), the equation specifying the curve can be written as:

$$y(t) = \sum_{i=1}^{n} C_i b_{i,4}(t) \quad \text{for} \quad x_4 \leq t \leq x_{n+1} \tag{5}$$

The B-spline basis allows one to write a particularly simple matrix equation when fitting splines to a set of data points. Suppose observations are made at m time points $t_1 \ldots t_m$ giving a vector $\mathbf{D} = (D_1, \ldots, D_m)^T$ of data points. We can then write the matrix equation $\mathbf{D} = \mathbf{SC}$, where $\mathbf{C}$ is the vector of n control points and $\mathbf{S}$ is a *m* by *n* matrix where $[S]_{ij} = b_{j,4}(t_i)$. If $n = m$ (the number of control points equals the number of data points), then $\mathbf{S}$ is square and the equation may be solved by matrix inversion, yielding an interpolating spline. However, as discussed this may lead to overfitting and it is often desirable to use fewer control points than data points to obtain a smoothing or approximating spline. In this case, the matrix equation must be solved in the least-squares sense, which yields $\mathbf{C} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{D}$.

3 Estimating Unobserved Expression Values and Time Points

In order to obtain a continuous time formulation, we use cubic B-splines to represent gene expression curves. As mentioned above, by knowing the value of the splines at a set of control points in the time-series, one

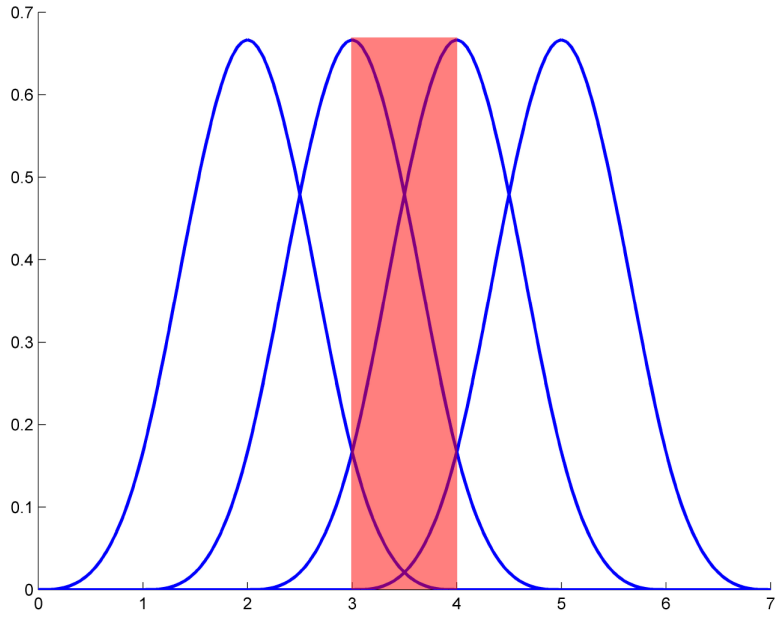


Figure 1: The B-spline basis functions are periodic (translates of one another) when a uniform knot vector is used. Shown here is the normalized cubic B-spline basis ($k = 4$) with knot vector $\mathbf{x} = (0, 1, 2, 3, 4, 5, 6, 7)^T$. Using the four basis functions shown, the B-spline will only be defined in the shaded region $3 \leq t \leq 4$, where all four overlap.

can generate the entire set of polynomials from the B-spline basis functions. In our formulation, the spline control points are uniformly spaced to cover the entire duration of the dataset. Once the spline polynomials are generated we can re-sample the curve to estimate expression values at any time-points. When estimating these splines from expression data, we do not try to fit each gene individually. Due to noise and missing values, such an approach could lead to over-fitting of the data and may in general lead to estimates that are very different from the real expression values of that gene (see Section 6.1). Instead, we constrain the spline coefficients of co-expressed genes to have the same covariance matrix, and thus we use other genes in the same class to estimate the missing values of a specific gene.

3.1 A Probabilistic Model of Time Series Expression Data

In this section we follow a method that is similar to the one used by James and Hastie [11] for classification. However, unlike their work, in this paper we focus on gene specific aspects rather than the common class profile. This allows us to handle variations in expression levels that are caused by gene specific behavior.

A class is a set of genes that are grouped together using prior biological knowledge or a clustering algorithm. In this section we assume that class information is given. We discuss how to deal with cases in which such class information is not given in Section 4.

We represent each gene expression profile by a spline curve. For a gene i in class j, $Y_i(t)$ is the observed value for i at time t. Let q be the number of spline control points used, and $s(t)$ the vector of spline basis

functions evaluated at time t, with $s(t)$ of dimensions q by 1. Denote by μ_j the average value of the spline coefficients for genes in class j, and by γ_i the **gene specific** variation coefficients. We assume that γ_i is normally distributed vector with mean zero and the class spline control points covariance matrix Γ_j, which is a q by q matrix. Denote by ϵ_i a random noise term that is normally distributed with mean 0 and variance σ^2. According to this model, $Y_i(t)$ can be written as:

$$Y_i(t) = s(t)(\mu_j + \gamma_i) + \epsilon_i$$

This model includes both gene specific and class specific parameters. This allows us to use information from other genes in the class based on the extent to which gene specific information is missing. We restrict the missing values of a gene by requiring them to vary with the observed values according to the class covariance matrix Γ_j. Using the class average μ_j and the gene specific variation γ_i, we can resample gene i at any time t' during the experiment. This is done by evaluating the spline basis at time t', and setting $\widehat{Y_i(t')} = s(t')(\mu_j + \gamma_i)$.

In order to learn the parameters of this model (μ, γ, Γ and σ) we use the observed values, and maximize the likelihood of the input data. Denote by Y_i the vector of observed expression values for gene i, and by S_i the spline basis function evaluated at the times in which values for gene i were observed. If we observed a total of m expression values for i in our dataset, then S_i is of dimensions m by q. The kth row in S_i contains the spline basis functions evaluated at t_k, where t_k is the time at which the kth value was observed. According to our model, we have:

$$Y_i = S_i(\mu_j + \gamma_i) + \epsilon_i$$

where ϵ_i is a vector of the noise terms. Note that since we are estimating the spline coefficients at the control points, each observed value has an effect related to the actual time it represents. Thus, different experiments can have different effects on the resulting curve if the expression values were sampled non-uniformly.

For our solution, we assume that the expression values for each gene were obtained independently of other genes. This assumption is not entirely true since different experimental conditions can affect multiple genes in the same experiment. However, this simplifying assumption allows for efficient computations and allows us to capture the essence of the results.

There are two normally distributed parameters in our model, the noise term ϵ and the gene specific parameters γ. Thus the combined covariance matrix for a gene in class j can be written as:

$$\Sigma_j = \sigma^2 I + S\Gamma_j S^T$$

where S is the spline basis function evaluated at all the time points in which experiments were carried out. Given this formulation, determining the maximum likelihood estimates for our model parameters is a non-convex optimization problem (see [11]). Thus, we turn to the EM algorithm. If the γ values were observed, we could have decomposed the probability in the following way:

$$p(Y, \gamma | \sigma^2, \Gamma, \mu) = p(Y | \sigma^2, \Gamma, \mu, \gamma) p(\gamma | \sigma^2, \Gamma, \mu)$$

which translates into the following joint probability:

$$\prod_j \prod_{i \in c_j} \frac{1}{(2\pi)^{n_i} \sigma^{n_i}} exp[-\frac{1}{2\sigma^2}(Y_i - S_i(\mu_j + \gamma_i))^T (Y_i - S_i(\mu_j + \gamma_i))] \times$$
$$\frac{1}{(2\pi)^q |\Gamma_j|^{1/2}} exp[-\frac{1}{2}\gamma_i^T \Gamma_j^{-1} \gamma_i] \quad (6)$$

where c is the number of classes and c_j is the set of genes in class j. Note that we need to maximize this joint probability simultaneously for all classes since the variance of the noise, σ^2, is assumed to be the same for all genes.

This representation leads to the following procedure. We treat the γ_i's as missing data and solve the maximum likelihood problem using the EM algorithm. In the E step we find the best estimation for γ using the values we have for σ^2, μ and Γ. In the M step we maximize equation 6 with respect to σ^2, μ and Γ while holding the γ_i's fixed. See [11] for complete details.

The complexity of each iteration of the EM algorithm is $O(q(n + c * q))$ since we estimate q parameters for each gene and $q^2 + q$ parameters for each class.

4 Model Based Clustering Algorithm for Temporal Data

The algorithm described in the previous section allows us to find the expression curve for each gene when the class partitioning is known. Though this information is sometimes available, either from previous knowledge or from a classification algorithm [16], this is not always the case. In this section we describe a new clustering algorithm that simultaneously solves the parameter estimation and class assignment problems.

TimeFit(Y, S, c, n) {
  For all classes j {
    choose a random gene i
    // initialize class center with a random gene
    $\mu_j = (S_i^T S_i)^{-1} S_i^T Y_i$
  }
  Initialize Γ, σ^2, γ arbitrarily
  Repeat until convergence {
    E step:
      for all genes i and classes j
        $p(j|i) \leftarrow \frac{p_j p(Y_i|\mu_j, \Gamma_j, \gamma_{i,j}, \sigma^2)}{\sum_k p_k p(Y_i|\mu_k, \Gamma_k, \gamma_{i,k}, \sigma^2)}$
    M step:
      for all genes i and classes j, Find the MAP estimate of $\gamma_{i,j}$
      // see Appendix for complete details
      Maximize Γ, σ^2, μ w.r.t. $P(j|i)$
      // update the class probability
      for all classes j, $p_j \leftarrow \frac{1}{n} \sum_i^n p(j|i)$
  }
}

Figure 2: Estimating the model parameters without class information. The posterior probabilities $P(j|i)$ can be used for clustering as described in the text.

Instead of the fixed class model from Section 3, we assume a mixture model. Thus, we can model the expression vector for gene i in the following way. First, we select a class j for gene i uniformly at random. Next, we sample γ_i using class j's covariance matrix Γ_j, and sample a noise vector ϵ_i using σ^2. Finally we construct Y_i by setting:

$$Y_i = S_i(\mu_j + \gamma_i) + \epsilon_i$$

In Figure 2 we present TimeFit, our spline fitting algorithm that performs class assignment. The number of desired classes, c, is an input to TimeFit. Initially all classes are assumed to have the same prior probabilities, though it is easy to modify this algorithm if one has prior knowledge about the different classes. TimeFit begins by choosing for each class one gene at random, and using this gene as an initial estimate for the class center (or average of the spline coefficients). We now treat the class assignments as the missing variables, and iterate using a modified EM algorithm. In the E step we estimate for each gene i and class j the probability that i belongs to class j, $P(j|i)$, using the values we obtained for the rest of the parameters in the M step. In the M step, instead of equation 6 we now maximize our parameters for each class with respect to the class probability ($P(j|i)$) as computed in the E step. In addition, we now treat the $\gamma_{i,j}$'s as parameters, and find their MAP (maximum a posteriori) estimate, which is then used in the E step. The complete details of this procedure are explained in the Appendix. As in the previous section, TimeFit still increases the likelihood monotonically, and terminates when the parameters converge.

When the algorithm converges, for each gene i we discover the class j such that $p(j|i) = max_{1 \leq k \leq c} P(k|i)$ and assign i to this class. Using this "hard" clustering, when we need to re-sample gene i's expression curve (either for missing values estimation or for new time points) we use the estimated class j's parameters ($\gamma_{i,j}, \mu_j$) and continue as described in the previous section.

For TimeFit, the complexity of each iteration of the EM algorithm is $O(cq(n+q))$ since we now estimate $c + cq$ parameters for each gene.

5 Aligning Temporal Data

The goal of the alignment algorithm is to warp the time-scale of one realization of a biological process into that of another. A set of genes is chosen in which the members are assumed to have the same temporal pattern of expression (e.g., from prior biological knowledge or clustering methods). A parameterized warping function is then selected and our algorithm seeks to produce an optimal alignment by adjusting the function parameters. Note that although it is possible to align individual genes this is problematic unless one has sufficiently high quality data as from replicates or a large number of time points.

Assume that we have two sets of time-series gene expression profiles, one of which we will refer to as the reference set. Denote a spline curve for gene i in the reference time series as $g_i^1(s)$, where $s_{min} \leq s \leq s_{max}$. Here, s_{min} and s_{max} are the starting and ending points for the reference time series respectively. Similarly, we will denote splines in the set to be warped as $g_i^2(t)$ for $t_{min} \leq t \leq t_{max}$. Define a mapping $T(s) = t$, which transforms points in the reference scale into the time-scale of the set to be warped. In this paper, we will discuss a linear transformation $T(s) = (s - b)/a$, with a the stretch/squash parameter and b the translation. However, more complex transformations could be used in our framework. We define the alignment error e_i^2 for each gene as:

$$e_i^2 = \frac{\int_{\alpha}^{\beta} [g_i^2(T(s)) - g_i^1(s)]^2 ds}{\beta - \alpha} \tag{7}$$

where $\alpha = \max\{s_{\min}, T^{-1}(t_{\min})\}$ is the starting point of the alignment, and $\beta = \min\{s_{\max}, T^{-1}(t_{\max})\}$ is its end point. The error of the alignment for each gene is proportional to the averaged squared distance between the curve for gene i in the reference set and in the set to be warped. In order to take into account the degree of overlap between the curves, and to avoid trivial solutions such as mapping all the values in the

curve to a single point, we divide this error by the time-length of the overlap $\beta - \alpha$. Thus, our goal is to find parameters a and b that minimize e_i^2. As discussed, we suggest minimizing the error for a set of genes. We define the error for a set of genes S of size n as:

$$E_S = \sum_{i=1}^{n} w_i e_i^2 \tag{8}$$

The w_i's are weighting coefficients that sum to one; they could be uniform ($1/n$) or used for unequal weighting. For instance, if one wishes to align wildtype time-series expression data with knockout data, many of the genes' expression patterns are expected to be unchanged in the two experiments. However, a subset of the genes may be highly affected. In this case, we want to down-weight the contribution of such genes, since they are not expected to align well. One way of formulating this is to require that the product $w_i e_i^2$ be the same for all genes (the weight will be inversely proportional to the error). From $w_i e_i^2 = K$, we get that $E_S = nK$ and so we can deduce that:

$$w_i = \frac{K}{e_i^2} \Rightarrow \sum_i w_i = \sum_i \frac{K}{e_i^2} \Rightarrow K = \frac{1}{\sum_i 1/e_i^2} = \frac{E_S}{n}$$

since $\sum_i w_i = 1$. As before, the objective is to minimize E_S, or in this case equivalently to maximize $\sum_i 1/e_i^2$.

Minimization of E_S must be done numerically, since a closed form solution is not possible. In the linear case presented, we are only searching for two parameters, so we minimize E_S directly using standard non-linear optimization techniques. We use the Nelder-Mead simplex search method (available in the Matlab package), which does not use gradients and can handle discontinuities. For the linear warping case, the essential constraints are that $\alpha < \beta$ and $a > 0$. Since the use of a numerical optimization method does not guarantee convergence to a global minimum, multiple random re-starts may be necessary. This leads to an algorithm with running time $O(rmnq^2)$, in which r is the number of random re-starts, m is the number of iterations for convergence, n is the number of genes in S, and q is the number of spline control points used.

If a large number of genes are to be aligned, we suggest the following algorithm to reduce the computation time. Begin by choosing a random subset of fixed size (e.g., 50 genes) and random initial settings for the warping parameters from a uniform distribution. The minimization procedure is then carried out and this process is repeated with a new random choice of warping parameters for a set number of iterations. Upon termination, the alignment parameters that correspond to the minimum error are chosen. These parameters are then used as the starting conditions for the E_S minimization using the full set of genes. See Section 6.2 for experimental results on how this reduces the running time on gene expression datasets.

6 Results

In this section we demonstrate the application of our method to expression time-series datasets, showing results for unobserved data estimation, clustering and alignment. Most of our results make use of the cell-cycle time-series data from Spellman $et\ al$ [16]. In that paper, the authors identify 800 genes in $Saccharomyces\ cerevisiae$ as cell cycle regulated. The authors assigned these genes to five groups that they refer to as $G1$, S, $S/G2$, $G2/M$, and $M/G1$.We also analyzed time-series data from a Fkh1/Fkh2 knockout experiment done by Zhu $et\ al$ [21]. Table 6 summarizes the data sets that we used.

dataset	method of arrest	start	end	sampling
alphaDS	alpha mating factor	0m	119m	every 7m
cdc15DS	temp. sensitive cdc15 mutant	10m	290m	ev. 20m for 1 hr, ev. 10m for 3 hr, ev. 20 min for final hr
cdc28DS	temp. sensitive cdc28 mutant	0m	160m	every 10m
Fkh1/Fkh2DS	alpha mating factor	0m	210m	ev. 15m until 165m, then after 45m

Table 1: Summary of gene expression time series analyzed.

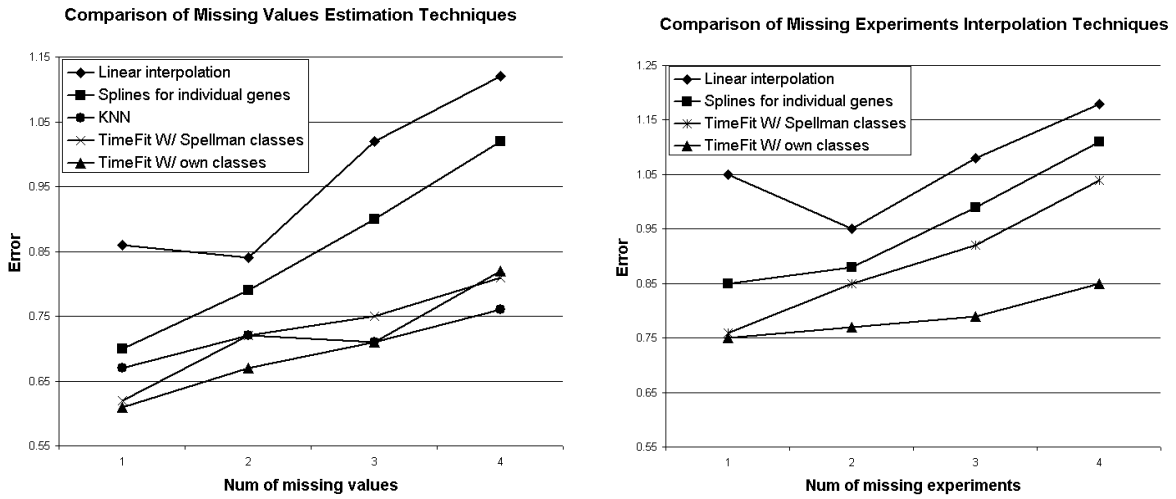
6.1 Unobserved Data Estimation

To test our missing value estimation algorithm we concentrated on the cdc15 dataset. We chose this dataset (see Table 6) because it is the largest (24 experiments) and contains non-uniformly sampled data. The results presented in this section were obtained using splines with 7 control points; however, similar results were obtained for different numbers of control points (results not shown).

We compared our algorithm to three other interpolation techniques that have been used in previous papers: linear interpolation [1], spline interpolation using individual genes [6], and k-nearest neighbors (KNN) with $k = 20$, which achieved the best results on static data out of all the algorithms described in [17]. In order to test our algorithm on a large scale we chose 100 genes at random from the set of cell-cycle regulated genes. For each of these genes we ran each estimation algorithm four different times, hiding 1,2,3 and 4 consecutive time points, while not altering the other genes. Next, we computed the error in our estimations when compared to an estimate of the variance of the log ratios of the expression values (see the Appendix for complete details).

Figure 3 (a) shows a comparison of the error of our estimation algorithm with the three methods mentioned above. For our method, we performed two separate runs. In the first we used the class information provided in [16] and in the second we used the algorithm described in Section 4 to obtain the class information. For one missing value, our algorithm achieves 10% less error than k-nearest neighbors (KNN). For two and three missing values our algorithm achieves lower or equal error rates when compared with KNN, and it does far better than the two other interpolation techniques. Only when trying to estimate four consecutive missing values does KNN perform better than our algorithm. However, four consecutive missing values are unusual in practice, and in almost all cases one does not need to estimate more than two consecutive values. Interestingly, our algorithm does better when it is allowed to estimate class membership than it does when the class information is pre-specified. This can be attributed to the fact that the five classes from [16] are somewhat arbitrary divisions of a continuous cycle. Thus, for missing value estimation our clustering algorithm is able to assign more relevant class labels.

Our algorithm can estimate expression values at any time point during the course of the experiments. In Figure 3 (b) we present results that were obtained by hiding 1,2,3 and 4 consecutive experiments. Again, our algorithm achieves more than 15% less error than the other two techniques. Note that KNN cannot be used to estimate missing experiments, and thus is not included in this comparison.



(a) Missing values (b) Missing experiments

Figure 3: Comparison among different missing value interpolation techniques. (a) Finding missing values and (b) finding missing experiments (time points not originally sampled). As can be seen, in almost all cases our algorithm does better than the others methods.

6.2 Clustering

In order to explore the effect that non-uniform sampling can have on clustering we generated two synthetic curves as follows. The first curve, f_1 is obtained using the equation $f_1(x) = \sin x$. The second curve is given by the following equation:

$$f_2(x) = \begin{cases} \sin x & : \quad x \leq \pi \\ \sin x + (x - \pi)/(20\pi) & : \quad x > \pi \end{cases}$$

We sampled each curve 64 times between $-\pi$ and π and then sampled between π and 5π (the remaining portion of the curve) at different rates of either every $\pi, \pi/2, \pi/4$ or $\pi/16$. Note that since all curves were sampled between $-\pi$ and 5π, the maximal difference between the sampled values (amplitude of the curves) is at most 0.2. For each different sampling we generated 100 vectors from each curve, and added random noise (normally distributed with mean 0 and variance 0.2). Next we used our TimeFit algorithm, and compared the results to those of k-means clustering. K-means is a clustering algorithm that assumes a mixture model and tries to assign genes to classes using the class centers (see [13] for details). K-means treats all points in the same way, and does not use the actual times they represent. As can be seen in Figure 4, the lower the sampling rate, the larger the difference between the performance of TimeFit and k-means. For example, for the sampling rate of π, k-means does only slightly better than chance, while TimeFit has a much higher classification success.

Next we tested TimeFit on the cdc15DS described above and compared the results to k-means. For both algorithms we generated five classes. When analyzing the results we used the Spellman clusters as the gold standard, and determined how many clusters in our results correspond to these clusters. The results are

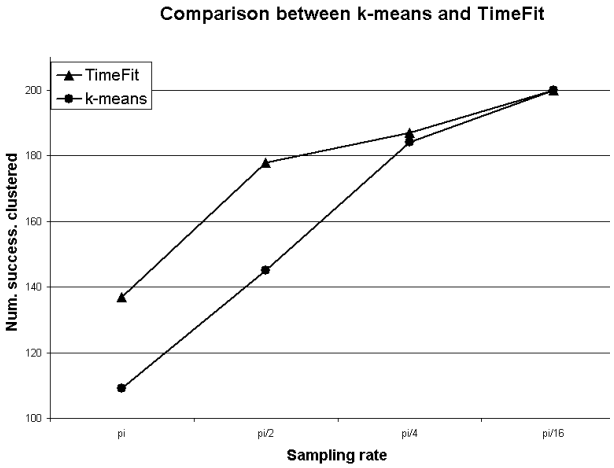


Figure 4: A comparison between k-means and TimeFit for clustering the vectors from f_1 and f_2. The success rate was determined by the total number of correctly clustered vectors out of the 200 vectors. As can be seen, the lower the sampling rate, the greater the advantage of using our algorithm.

presented in Table 2. As can be seen, four out of the five clusters that were generated by TimeFit correspond to Spellmans' clusters, containing at most two neighboring phases with 10 or more genes (the fifth contained genes from three consecutive phases). Since we are dealing with cell cycle data, the clusters defined in [16] can only have arbitrary boundaries and thus joining two of them is reasonable. On the other hand, in the k-means clustering result, cluster 4 contains more than 20 genes from four different phases, while cluster 5 contains more than 20 genes from three different phases. Thus, the results of our clustering algorithm are in better correspondence with existing biological knowledge than those of k-means.

	TimeFit					k-means				
Phase / Cluster	1	2	3	4	5	1	2	3	4	5
G2/M	121	52	4	0	8	117	6	1	60	1
M/G1	2	62	33	8	1	6	51	3	45	1
G1	0	9	166	40	71	0	24	125	104	33
S	0	1	2	0	65	0	0	3	7	58
S/G2	30	3	4	0	81	18	0	0	22	78

Table 2: Comparison between k-means and our clustering algorithm on the cdc15DS. As can be seen, in most cases each of the clusters generated by TimeFit contains genes from 1 or 2 neighboring phases. For k-means there is one cluster (4) that contains more than 20 genes from 4 different phases.

6.3 Alignment

We aligned three yeast cell-cycle gene expression time-series that clearly occur on different time-scales and begin in different phases. The cdc15DS was used as a reference set and the alphaDS and cdc28DS were aligned against it using a linear warping $T(s) = (s - b)/a$ and the full set of cell-cycle regulated genes as identified in [16]. For the cdc28DS, we obtained $a = 1.42$ and $b = 2.25$ with $E_S = 0.1850$. These results indicate that the cdc28DS cell-cycle runs at approximately 1.4 times the speed of the cdc15DS cycle and starts approximately 5.5 minutes before (i.e., we calculate $T(10)$ since cdc15DS starts at 10 minutes). For the alphaDS, we obtained $a = 1.95$ and $b = -5.89$ with $E_S = 0.1812$. Figure 5 shows the aligned/unaligned expression values for the $G1$ and $S/G2$ clusters for the cdc28DS to cdc15DS alignment. Alignment for each dataset took approximately 5.5 minutes on a 1 GHz Pentium III machine using our algorithm that performs initial alignments on smaller subsets of genes; the alignments took approximately 45 minutes without this improvement.

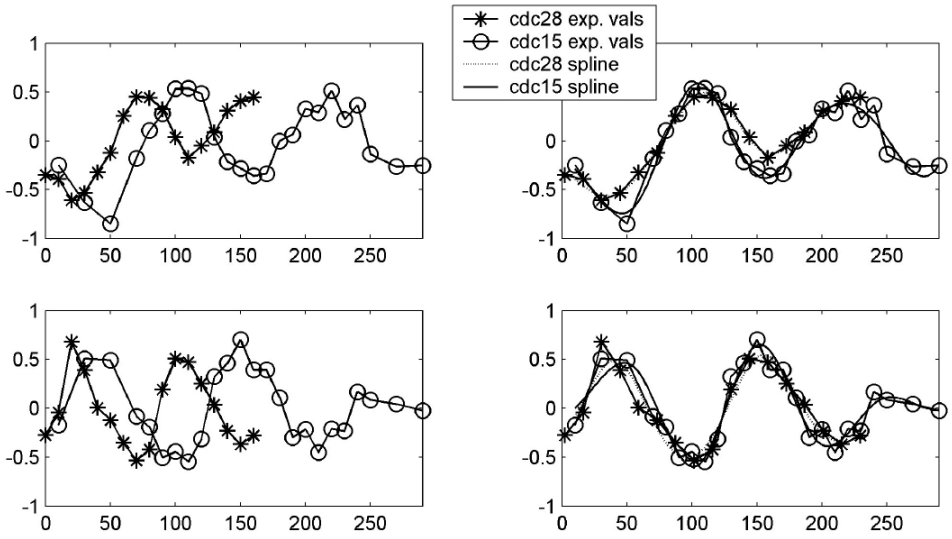


Figure 5: Alignment of genes for the cdc28DS to cdc15DS. Linear warping was used with the full set of cell-cycle regulated genes. The left-hand side shows class-averages of unaligned expression values for two clusters. The top row shows aligned results for the $G1$ cluster (186 genes) and the bottom row the $S/G2$ cluster (283 genes). These results indicate that the cdc28DS cell-cycle runs at approximately 1.4 times the speed of the cdc15DS cycle and starts approximately 5.5 minutes before.

To validate the quality of these alignments we performed two analyses: 1) alignments of genes in alphaDS against genes in cdc15DS with gene identity of those in cdc15DS randomly permuted and 2) alignments of alphaDS to cdc15DS using different numbers of genes. Note that for brevity only the alphaDS was used; we chose this dataset because it is the smallest and presumably demonstrates worst-case results. For the first analysis, we performed 200 trials giving E_S scores between 0.2562-0.3554, with 50% of the scores lying between 0.2780-0.3129. These results suggest that the actual alphaDS to cdc15DS E_S score of 0.1812 would not arise by a chance alignment of the genes. For the second analysis, we sampled subsets of

# genes	*a*	std *a*	*b*	std *b*
5	1.80	0.42	-7.70	16.41
10	1.89	0.21	-5.06	21.5
25	1.93	0.10	-4.99	7.07
50	1.93	0.13	-5.13	9.03
100	1.96	0.03	-6.86	2.42
200	1.95	0.02	-6.38	1.76
400	1.96	0.02	-6.12	1.30

Table 3: Results of experiments in which random subsets of fixed size were sampled 100 times and alignment of alphaDS and cdc15DS were performed. The columns are as follows: number of genes used, stretch/squash parameter, standard deviation of this parameter, offset parameter, and standard deviation of this parameter. This analysis shows that the variance in the parameters decreases as more genes are used and there is convergence to the a and b settings found with the full set of genes.

between 5-400 genes 100 times from the full set of cell-cycle regulated genes (Table 3). This analysis shows that the variance in the parameters decreases as more genes are used and there is convergence to the a and b settings found with the full set of genes. Interestingly, our algorithm is usually able to find the "actual" a and b parameter settings even when relatively small numbers of genes are used.

Thus, these analyses give evidence that our algorithm can reliably align the cell-cycle datasets. These results compare favorably with those in Aach $et\ al$ [1] using the same data. In their case, they found that their actual alignment score was not at a low percentile when compared against alignments using randomized data (gene values shuffled). Further, they indicate that poor results were obtained with small cluster sizes (an analysis over a wide range of sizes was not presented in their paper). The fact that our method uses a continuous representation and fits only two parameters to all the genes helps to explain its good performance on the cell-cycle data. However, one must be careful in extrapolating these results, since they are clearly dependent on the underlying dataset.

In a second application of our alignment algorithm, we used our method to discover yeast cell-cycle regulated genes that appear to be regulated by the Fkh2 transcriptional factor. Zhu $et\ al$ performed an experiment in which two yeast transcriptional factors (Fkh1 and Fkh2) were knocked out and a time-series of gene expression levels was measured in synchronized cells [21]. Simon $et\ al$ [15] demonstrated with a microarray DNA-binding experiment that a set of genes are bound by Fkh2 in wildtype unsynchronized yeast. We were interested in discovering which genes in this set show altered expression in the knockout experiments. However, direct comparison of the data from Zhu $et\ al$ [21] and that from Spellman $et\ al$ [16] is problematic, because the series were sampled at different rates, begin at different cell-cycle phases, and exhibit different periods.

We used our algorithm to rank fifty-six genes bound by Fkh2 according to the difference in expression curves of the aligned wildtype and knockout experiments. The non-uniform weighting version of our algorithm was used to align the datasets using all genes identified as cell-cycle regulated in [16] and the gene alignment error scores e_i^2 were used for ranking. Figure 6 shows a plot of the spline expression profiles of the top four genes with the worst alignment scores and the top four with the best scores. A poor alignment score indicates that a gene is behaving differently in the knockout experiment.

The ranking produced by our algorithm appears to yield biologically meaningful results, highlighting

which genes appear to be regulated by Fkh2 and those that are merely bound by it. For instance, all of the genes with the worst alignment scores shown were determined to be bound by both Fkh1 and Fkh2 in [15], whereas all of the best aligning genes were determined to be bound by Fkh2 only. This corresponds to biological knowledge indicating that both Fkh1 and Fkh2 are required for regulation of a number of genes. It is also interesting that among the genes shown with good alignment, three are bound also by Swi6 and either Mbp1 or Swi4, factors that are likely to work independently of the Fkh proteins. Further, the genes with poor alignment are known to be bound by Ndd1 and Mcm1 or Ace2 and/or Swi5. Mcm1/Ndd1 are known to work with the Fkh proteins and are not sufficient to regulate expression without them. Ace2 and Swi5 apparently can bind and regulate independently of the Fkh proteins, but their expression is Fkh dependent.

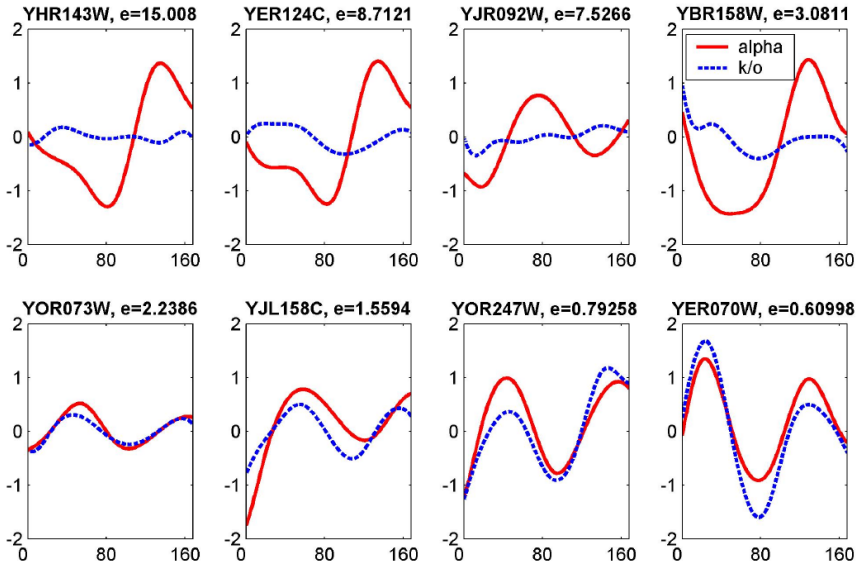


Figure 6: Alignment of Fkh1/Fkh2 knockout data and the wildtype alphaDS. Genes shown are from a set of genes demonstrated to be bound by Fkh2. Shown are the genes with the four worst (top row) and best (bottom row) gene alignment scores. A poor alignment score indicates that a gene is behaving differently in the knockout experiment. See text for biological interpretation of these results.

7 Conclusion and Future Work

We presented a unified model and algorithms that use statistical spline estimation to represent gene time-series expression profiles as continuous curves. Results using our approach on a large yeast cell-cycle data set demonstrate that our framework, when used for estimating unobserved time-points, clustering, and alignment of datasets has substantial advantages over other methods that treat time-series as vectors of points. Overall, we believe that as the analysis of dynamic genetic behavior becomes more sophisticated, principled model-based methods such as ours will become essential for reconstructing and combining data.

There are a number of interesting extensions that could be made to our work. Experimental biologists often determine the sampling rate for a time-series experiment based on knowledge about how quickly gene expression values change. These assessments often make little use of information that may be gleaned from previous expression experiments. Our algorithm could be used to find the ”right” sampling rate for time-series experiments, which could lead to substantial time/cost savings or improvements in biological results. Another way of extending this work is to develop a clustering algorithm that uses our alignment method in order to group genes that show similar kinetic changes between datasets. Another open problem is developing a principled method for determining the significance of the alignment error in order to automatically detect genes whose temporal behavior is altered between experiments.

Acknowledgements

Z.B.J. is supported by a Fellowship from the Program in Mathematics and Molecular Biology at the Florida State University, with funding from the Burroughs Wellcome Fund Interfaces Program.

References

[1] J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17:495–508, 2001.

[2] R. Bartels, J. Beatty, and B. Barsky. *Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufman, 1987.

[3] B. Brumback and J. Rice. Smoothing spline models for the analysis of nested and crossed samples of curves. *Am. Statist. Assoc.*, 93:961–976, 1998.

[4] S. Chu, J. DeRisi, and et al. The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705, 1998.

[5] L. Deng, M. Aksmanovic, D. X. Sun, and C. F. J. X. Wu. Recognition using hidden markov models with polynomial regression functions as nonstationary states. *IEEE Transactions on Speech and Audio Processing*, 2:507–520, 1994.

[6] P. D’haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. In *PSB99*, 1999.

[7] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[8] R. Eubank. *Nonparametric regression and spline smoothing*. Marcel Dekker, 1999.

[9] N. Friedman, M. Linial, I. Nachman, and D. Pe’er. Using bayesian network to analyze expression data. In *RECOMB*, 2000.

[10] N. S. Holter, A. Maritan, and et al. Dynamic modeling of gene expression data. *PNAS*, 98:1693–1698, 2001.

[11] G. James and T. Hastie. Functional linear discriminant analysis for irregularly sampled curves. *Journal of the Royal Statistical Society*, to appear, 2001.

[12] S. H Neal, M. Madhusmita, and et al. Fundamental patterns underlying gene expression profiles: Simplicity from complexity. *PNAS*, 97:8409–8414, 2000.

[13] Sharan R. and Shamir R. Algorithmic approaches to clustering gene expression data. *Current Topics in Computational Biology*, To appear.

[14] D. Rogers and J. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, 1990.

[15] I. Simon, J. Barnett, and et al. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[16] T. S. Spellman, G Sherlock, and et al. Comprehensive identification of cell cycle-regulated genes of the yeast $saccharomyces\ cerevisia$ by microarray hybridization. *Mol. Biol. of the Cell*, 9:3273–3297, 1998.

[17] O. Troyanskaya, M. Cantor, and et al. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 2001.

[18] P. Viola. *Alignment by Maximization of Mutual Information*. PhD thesis, MIT AI Lab, 1995.

[19] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.

[20] L. P. Zhao, R. Prentice, and L. Breeden. Statistical modeling of large microarray data sets to identify stimulus-response profiles. *PNAS*, 98:5631–5636, 2001.

[21] G. Zhu, Spellman T. S., and et al. Two yeast forkhead genes regulate cell cycle and pseudohyphal growth. *Nature*, 406:90–94, 2000.

A EM Algorithm for Class Assignment

In this appendix we present the details of the EM algorithm that is used in Section 4. We start with the complete log likelihood given by:

$$\sum_i \log(\sum_j Z(j|i)\frac{1}{\sigma^{n_i}} exp[-(Y_i - S_i(\mu_j + \gamma_{i,j}))^T (Y_i - S_i(\mu_j + \gamma_{i,j}))/2\sigma^2] \times \frac{1}{|\Gamma_j|^{1/2}} exp[-\frac{1}{2}\gamma_{i,j}^T \Gamma_j^{-1} \gamma_{i,j}]) \quad (9)$$

where j is the class index and n_i is the number of observed values for gene i. $Z(j|i)$ is an (unobserved) binary indicator variable that assigns each gene to exactly one class.

In the E step we compute the expected values for $Z(j|i)$:

$$p(j|i) = E(Z(j|i)|Y_i) = \frac{p_j e^{-(Y_i - S_i(\mu_j+\gamma_{i,j}))^T (Y_i - S_i(\mu_j+\gamma_{i,j}))/\sigma^2} e^{-\frac{1}{2}\gamma_{i,j}^T \Gamma_j^{-1} \gamma_{i,j}}}{\sum_k p_k e^{-(Y_i - S_i(\mu_k+\gamma_{i,k}))^T (Y_i - S_i(\mu_k+\gamma_{i,k}))/\sigma^2} e^{-\frac{1}{2}\gamma_{i,k}^T \Gamma_k^{-1} \gamma_{i,k}}}$$

In the M step we first find the MAP estimate for $\gamma_{i,j}$ by setting:

$$\gamma_{i,j} = (\sigma^2 \Gamma_j^{-1} + S_i^T S_i)^{-1} S_i^T (Y_i - S_i \mu_j)$$

Next, we maximize σ^2, μ and Γ w.r.t. the class assignment probabilities computed in the E step:

$$\sigma^2 = \frac{\sum_i \sum_j p(j|i)(Y_i - S_i(\mu_j + \gamma_{i,j}))^T (Y_i - S_i(\mu_j + \gamma_{i,j} + trace((\Gamma_j^{-1} + S_i^T S_i)^{-1} + S_i^T S_i))}{\sum_i n_i}$$

μ_j is computed by setting:

$$\mu_j = (\sum_i p(j|i) S_i^T S_i)^{-1} (\sum_i p(j|i) S_i^T (Y_i - S_i \gamma_{i,j}))$$

Then we set Γ_j to:

$$\Gamma_j = \frac{\sum_i p(j|i)[\gamma_{i,j} \gamma_{i,j}^T + (\hat{\Gamma}_j{}^{-1} + S_i^T S_i / \sigma^2)^{-1}]}{\sum_i p(j|i)}$$

B Computing Error Rates

Here we describe in detail the method we used to compute the error rates of the four different missing values algorithms discussed in Section 6.1. Denote by $Y_i(t)$ the (hidden) expression values for gene i at time t, and by $\widehat{Y_i(t)}$ the estimated values. Denote by m the number of missing (hidden) data points and by n the number of genes that were used for the test. Denote by v the variance of the log ratios of expression values. Then the error of an estimation for m missing data points is defined as:

$$err_m = \frac{1}{mn} \sum_{i=1}^{n} \sum_{l=1}^{m} \sqrt{\frac{[Y_i(t_l) - \widehat{Y_i(t_l)}]^2}{v}}$$

If err_m is above 1 then the error is (on average) bigger than the replication variance, and vice versa. The variance v was computed using the raw expression data of the unsynchronized cells from two different time points.