CLASSIFICATION AND ALIGNMENT OF GENE-EXPRESSION TIME-SERIES DATA

by

Adam Allen Smith

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2009

לעתיד.

ABSTRACT

We present methods for comparing and performing similarity queries for gene-expression time-series data. Such data is usually gathered via microarrays or related technologies. In the studies with which we work, the methods are used to compare the gene activity of mice after exposure to different treatments, or with specific genes knocked out. This lets us compare the effects of the treatments or knockout at a molecular level. The data tends to be sparse in time, but it represents measurements for thousands or tens of thousands of separate genes, each of which constitutes a separate dimension. Such data is also subject to technical noise and biological variability.

Our approach involves three key steps. The first step is to reconstruct a continuous time series from the discrete observations. We use B-splines to accomplish this. Unlike previous methods, we relax the fit of the splines so that they are less prone to overfitting the data. We place the points of discontinuity in the spline in such a way that a spline is well-defined over the whole length of the series.

The second step is to align the pairs of time series in order to find a time-by-time correspondence that maximizes the similarity between them. We present two segment-based algorithms that are specially designed to align gene-expression data. We also develop heuristics to speed up the alignment computations, without adversely affecting the quality of the alignments found. Finally, we present an approach for computing clustered alignments, in which the genes are split into a small number of clusters, each of which is aligned independently.

The final step is to score the alignments found, based on the similarity of the two series. This allows us to conduct similarity searches, in which we compare a query of unknown character to series associated with other treatments that have been well-studied. One of our high-level goals is to create a BLAST-like tool, that will allow biologists to enter the gene-expression data from their own studies, and will return treatments that affect gene expression in similar ways.

ACKNOWLEDGMENTS

A dissertation cannot be completed in a vacuum. There are many people who have helped me along the way over the last years, whom I would like to thank.

Thanks first go to my advisor, Mark Craven. He offered me a position when he barely knew me, and his faith in me through the years is one of the primary reasons that I was able to see this degree to its conclusion. He was always patient and helpful, and has a truly subtle sense of humor that I appreciated. For these reasons I will always be grateful.

Our EDGE collaborators Christopher Bradfield, Aaron Vollrath, and Kevin Hayes were also a big source of help. Aaron in particular was able to keep a level head as I e-mailed him many times after midnight with a frantic biology question. Your help was much appreciated.

David Page, Jude Shavlik, and Jignesh Patel rounded out my committee (which also includes Mark and Chris). Amos Ron also assisted as part of my preliminary exam committee. Thank you for taking the time to read through my work.

My family has also been a constant source of support through the years, even as they have tried to understand this arcana. Mom and Bob and Steve and Kelsey, I love you. My cousins Jordan Hiller and Jenny Stadler inspired me to think about grad school when I was still a teenager. (I am returning the favor on their kids, slowly.) Susie and Ted Johnson, Neil and Barbara Kristiansson, Fred and Judith Lothrop, April and Gary Thompson, and Lucas and Hayla Thompson also deserve thanks for their support and hospitality. I would also like to mention my grandmothers, LaVonne Smith and Areta Stadler. Both saw me start grad school, but unfortunately neither of them will see me finish. Areta in particular mentioned that she was eager to read my dissertation.

My funding has come from the National Institues of Health, specifically the National Institute of Environmental Health Sciences, the National Library of Medicine, and the National Cancer Institute.

Finally, there are numerous other people who have helped me along in various ways. Some helped with my research by pointing out small insights that I had missed. Others helped through their friendship, or by giving me a place to sleep during my frequent travels. Among these are Dave Andrzejewski, Drew Bernat, Jenn Bernat, Debbie Chasman, Kathy Finkle, Kevin Finkle, Sigrun Franzen, James Frey, Allison Holloway, Erin Jonaitis, Graham Jonaitis, Elizabeth Kenniston, Sean McIlwain, Deborah Muganda, Keith Noto, Louis Oliphant, Irene Ong, Yue Pan, Soumya Ray, Christine Reilly, Burr Settles, Herschel Snodgrass, Lisa Torrey, Michael Wallick, Hao Wang, and Ursula Whitcher. Thank you guys.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

NOMENCLATURE

COW	correlation-optimized warping
$\vec{d}$	database time series
DNA	deoxyribonucleic acid
DTW	dynamic time warping
$\mathbf{\Gamma}$	warping matrix
$\gamma(i,j)$	element of warping matrix
mRNA	messenger RNA
miRNA	micro RNA
PTW	parametric time warping
RNA	ribonucleic acid
SCOW	shorting correlation-optimized warping
$\vec{q}$	query time series
TCDD	2,3,7,8-tetrachlorodibenzo-*p*-dioxin
tRNA	transfer RNA

Chapter 1

Introduction

The comparison and alignment of gene-expression time-series data is an important problem in modern computational biology. Figure 1.1 shows a small example of this kind of data. We have several sets of data (*treatments*, in our domain), each of which consists of multiple channels of data

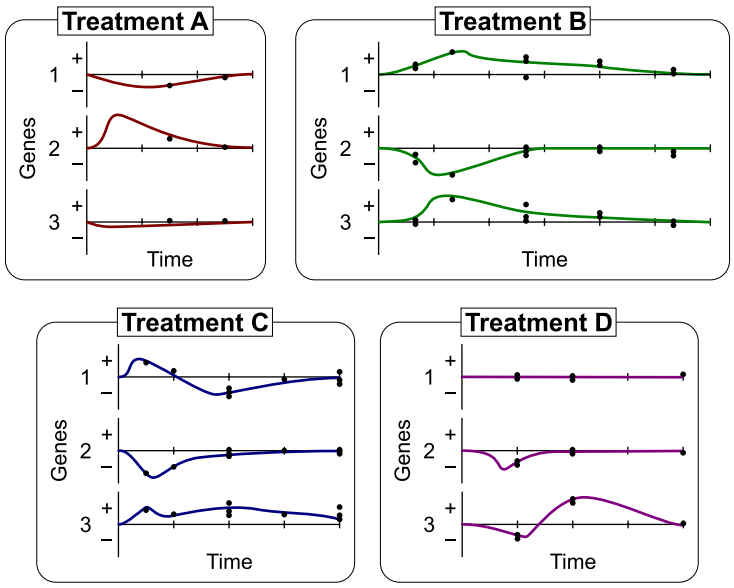

Figure 1.1: Toy examples of gene-expression time-series data. Here four different time series, each from a different *treatment*, or experimental condition, have data for three genes. The points indicate discrete observations of the underlying gene expression levels. Real data contains measurements for thousands of genes.

(*genes*) whose values vary over time. We have developed algorithms that improve the state of the art in comparing such data to find their similarities and differences.

Roughly speaking, a gene is a stretch of DNA within a cell that encodes a product, such as an enzyme or other protein, or an RNA molecule. (We will define "gene" formally and in more depth in Chapter 2.) A gene that is being read to create its product is said to be being "expressed." Because cells have limited resources, cells will generally only express genes when they are needed. Expression varies between cells based on many factors, including outside stimuli, expression levels of other genes, signals like hormones or enzymes, and cell type (in multicellular organisms). Obtaining a snapshot of a cell's gene-expression values gives us an informative but partial description of the cell's state.

Because genes are not static, their activities are often best represented as a time series. However, at present there does not exist any technology to cheaply and accurately observe the gene-expression levels of a cell continuously. We must make do with making observations at several discrete times in order to estimate the underlying activity. This problem is illustrated in Figure 1.1. The curves represent the hidden expression levels of the genes as they vary over time, while the dark points represent discrete observations.

Numeous biological studies have collected gene-expression time series. Here we consider a few examples. One of the most prominent examples is that of yeast (*S. cerevisiae*) cell-cycle data (Spellman et al., 1998). The expression levels of genes in yeast rise and fall as the cells go through their reproductive cycle. Spellman et al. used various methods to synchronize yeast cell populations, and then measured expression levels at multiple times during the cell cycle. Aach and Church (2001) were the first to compare these series, using the standard dynamic programming techniques similar to those we discuss in this paper.

Khodursky et al. (2000) measured gene-expression time series in *E. coli* in order to characterize the genes regulating the bacteria's synthesis of the amino acid tryptophan, in part by observing gene activities over time when the bacteria were exposed to differing external amounts. More recently, Ong et al. (2002) applied dynamic Bayes networks to this data in order to model the changes in gene-expression levels over time.

Circadian rhythm studies have also produced gene-expression time-series data. Certain genes are strongly associated with the circadian cycle in a vareity of organisms. Storch et al. (2002) gathered data on these genes from the hearts and livers of mice over a 48 hour period. Comparing the data from the two organs, they found little overlap between the expressed circadian genes in the heart, and those in the liver. In a separate study, Claridge-Chang et al. (2001) gathered time-series data on circadian genes from the heads of fruit flies (*D. melanogaster*) over six days. They isolated several genes showing strong circadian cyclical expression patterns.

Several research groups have also gathered gene-expression time-series data from embryonic stem cells, in order to better understand their pluripotent properties. Chang et al. (2006) detailed methods used to gather data on human embryonic stem cells. Chen et al. (2007) focused on gene-expression time series which are related to the development of the circulatory system. Tuke et al. (2009) applied some of the same techniques to mouse embryonic stem cells. They observed the cells in the process of differentiation over the course of nine days, in order to pick out the ones associated with differentiation.

1.1 Time Series Comparison Subtasks

We break down the comparison of time series into three main steps:

i. *Reconstruction*, in which we use interpolation techniques to fill in missing data, so that the time series we are aligning appear to be regularly sampled.

ii. *Alignment*, in which we identify a mapping from time coordinates in one series to the time coordinates in another series such that the similarities in their expression responses are maximized.

iii. *Similarity Search*, in which we quantitatively assess how alike two series are, allowing us to identify the time series in a database that is most similar to a given query.

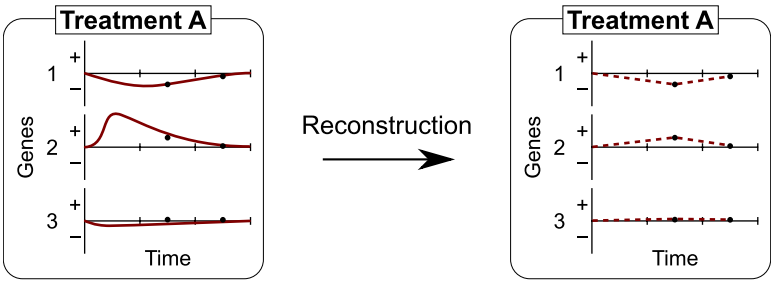

Figure 1.2: Toy gene-expression reconstruction problem. Only a limited number of observations are made, represented by the dark points. Expression levels at intermediate times must be reconstructed from what data is available.

1.1.1 Reconstruction

Figure 1.2 illustrates the problem of observations only being made at discrete times. Often these observations are sparse in time, and will be taken at different times in different treatments. In order to compare time series, we first reconstruct the data so that we can compare regularly-sampled series.

Early studies used linear interpolation (Aach and Church, 2001) to reconstruct intermediate time points. However there has been success in using more complicated interpolation methods, such as B-splines (Bar-Joseph et al., 2003; Luan and Li, 2004). We have developed methods that use B-splines not only to interpolate missing data, but also to filter out extreme readings from the data.

1.1.2 Alignment

The next step in comparing two time series is to align them, as shown in Figure 1.3. Naïvely comparing the same time points in two series will often not reveal the degree of similarity of the series (Keogh and Pazzani, 2000; Aach and Church, 2001). The similarity of two series can be greatly underestimated if there is biological variation in the temporal response, error in observation times, or if the series are even slightly out of phase with one another. In the process of alignment, one searches for a mapping between the two series that maximizes their similarity. This is often

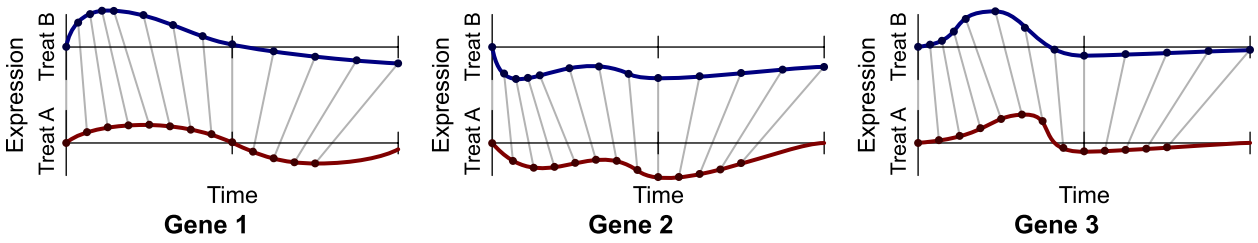

Figure 1.3: Toy gene-expression alignment problem. The lines joining the expression curves show the best mappings from times in one gene to times in the other. This alignment is non-global, as the end of the bottom series does not correspond to any part of the top one.

called *time warping* or just *warping*, because it is equivalent to subtley distorting one of the series so that the two line up. Often, some limit is placed on the warping to prevent very dissimilar series from appearing too much alike. For example, we could exclude warpings in which the absolute difference in times is more than some threshold.

It is often not clear whether the entirety of two series being compared should be accounted for by the alignment. Consider the case in which we apply a chemical treatment to an organism. There is a well-defined starting point at which we can begin to monitor the effects of the treatment, but it is difficult to know when and if these effects have finished. In other cases, there may not even be a well-defined starting point, making the alignment much harder. The majority of alignment algorithms perform global alignments. These algorithms make the implicit assumption that both series have advanced to the same point by the end of their observations. By contrast, we have devised non-global methods of alignment, such as that illustrated in Figure 1.3.

Common time warping methods include parametric time warping (Eilers, 2004), dynamic time warping (Keogh and Pazzani, 2000), and segment-based time warping (Nielsen et al., 1998). The algorithms differ in speed and accuracy, and have different strengths in different domains. We have developed a pair of methods, *multisegment generative* and *shorting correlation-optimized warping* (*SCOW*), which we detail in Chapter 5.

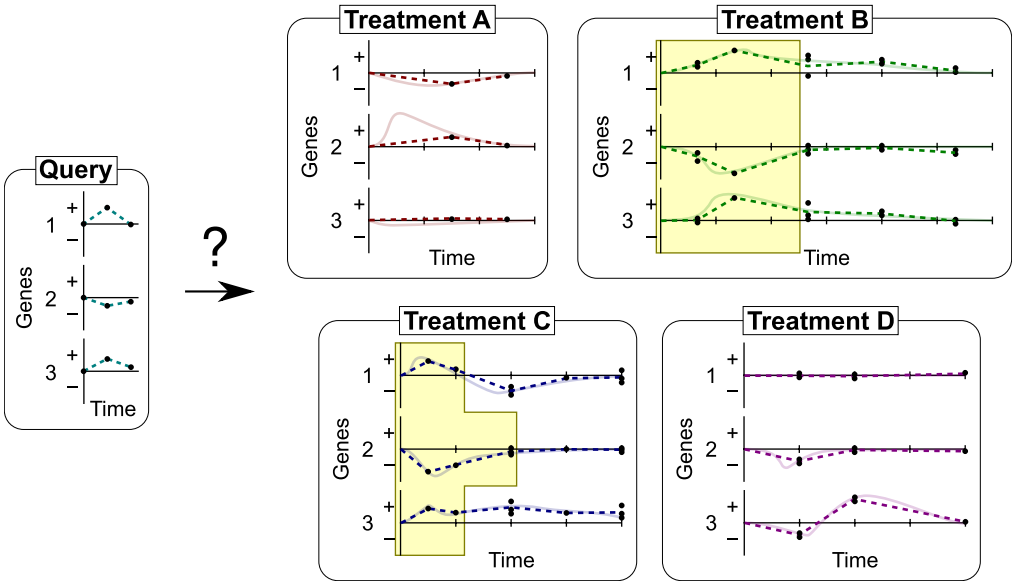

Figure 1.4: Toy gene-expression similarity problem. Given a query, we want to determine to which previously seen treatment it is most similar. Here, the highlighted areas within Treatments B and C show strong similarity. In Treatment B, all the genes have been warped together. In Treatment C, they have been warped separately.

1.1.3 Similarity Search

The final step is to score the alignment between the two series. Scoring pairs of series allows us to perform a similarity search for a query time series among a database of labeled or previously characterized series. This task is illustrated in Figure 1.4. Here, either Treatment B or Treatment C might be good matches for the query series. We discuss our scoring methods in Chapter 5 and consider the task of similarity search in Chapters 5, 6, and 7.

1.2 Motivations

One of our chief goals has been to create a tool analogous to the *Basic Alignment Search Tool* (BLAST), but for time-series data. BLAST is an algorithm that compares biological sequences such as nucleotides or amino-acid sequences (Altschul et al., 1990). By doing a BLAST search, a

researcher can compare a query sequence to a database of other sequences, and identify those that resemble the query. The methods we have developed allow similar queries about gene-expression time-series data. This task is illustrated in Figure 1.4.

The development of such a tool helps in the need for faster, more cost-efficient protocols for characterizing the potential toxicity of industrial chemicals. More than 80,000 chemicals are used commercially, and approximately 2,000 new ones are added each year (Hayes et al., 2005). This number makes it impossible to properly assess the toxicity of each compound in a timely manner using conventional methods. However, the effects of toxic chemicals may often be predicted by how they influence gene expression. It is likely that gene-expression profiles will soon become a standard component of toxicology assessment and government regulation of drugs and other chemicals (Thomas et al., 2001).

For example, let us say that we have a new chemical that needs to be characterized. We have a database of time series, in which the expression levels of a certain set of mouse genes are increased after exposing the animals to an inflammatory agent, but decreased after exposure to a hypoxic agent (one which prevents tissues from getting enough oxygen). We can first obtain gene-expression measurements after exposing mice to the new chemical, and then compare these measurements to those in the database using our algorithms. Based on this, we might determine if the chemical has inflammatory or hypoxic properties, and can then decide on further study in this direction.

Another motivation has been the comparison of organisms having a particular gene "knocked out" or disabled, with wildtype organisms with the functioning gene. By comparing gene-expression time-series data between these two groups, one can elucidate the role of that gene in the organism's genetic regulatory network (i.e. the complex system of genes that influence one another's expression). Our work has focused on Mop3, which is one of the central regulating genes of the circadian cycle in mice (Bunger et al., 2000, 2005). Knocking out Mop3 causes a cascade of other effects in other genes. We believe that genes that are aligned in similar ways between the wildtype and knockout profiles may be closely related in the network.

1.3 Special Considerations for Gene-Expression Time Series

When comparing two gene-expression time series, there are several factors that we must consider. We will revisit these often throughout this work.

- *Discrete observations:* At present, there does not exist any well-developed technology to continuously monitor genes within a living cell. We must make many separate observations at discrete times, and reconstruct the hypothetical gene-expression levels at intermediate times that were not observed.

- *Temporal sparsity:* Most time series available from gene-expression studies contain only a handful of time points (Ernst et al., 2005). This makes reconstruction of a continuous time series difficult.

- *High-dimensionality:* Current methods of collecting gene-expression data allow us to measure thousands of genes simultaneously. Thus each time "point" in our time series lies in a high-dimensional space. The data sets we explore represent thousands or tens of thousands of separate genes.

- *Non-uniform and irregular sampling:* Given the sparsity of the time series, it is typically the case that they have been sampled at non-uniform time intervals. Moreover, the sampling times may vary for different time series.

- *Noise:* Current data collection methods suffer from technical limitations that introduce noise into the data. Fortunately, methods are improving that will minimize this noise in the future. However, at present it is a problem that we must address.

- *Biological variability:* When gene-expression experiments use an animal model, there is also a component of biological variation that affects the data measured. In some cases, each data point is sampled from a different animal. Moreover, the animals may be genetically distinct.

These properties of time-series data result in several additional challenges when aligning a pair of time series.

- The time points present in one series may not correspond to measured points in the other.
- The series may be of different size. They could consist of a single observation, or many. Additionally, they may vary in their extent: one could span only a few minutes or hours while the other could include measurements taken over days.
- Series may differ in amplitude, temporal offset, or temporal extent of their responses. For example, two series may be similar, except that the gene-expression responses in one are attenuated, occur later, or take place more slowly.
- Two series may differ in that one of them shows more temporal evolution. In other words, one series may appear to be a truncated version of the other.

1.4 Open Problems

In comparing two time series, there are several open problems that we have attempted to answer:

- How can one accurately reconstruct time series when there are irregular observations and noise? Previous methods have mostly used linear interpolation to reconstruct data (e.g. Aach and Church (2001)), or aligned the unreconstructed data directly (e.g. Ernst et al. (2005)). Bar-Joseph et al. (2003) proposed the use of B-splines for interpolation. However, the method that they use has two principal shortcomings. First, the splines they calculate may be undefined if there is a large enough gap between successive observations. Second, B-splines can often overfit the data. In such a case, the "interpolating" curves will oscillate greatly between the observed points. Our contribution has been to develop a fitting algorithm that will return well-defined splines regardless of time between observations, and that will filter out extreme high and low expression levels. We cover this approach in Chapter 4.

- Previous approaches to aligning gene-expression time-series data have used parametric time warping (PTW) (Bar-Joseph et al., 2003) or dynamic time warping (DTW) (Aach and Church, 2001). As shown in greater detail in Chapter 2, both of the methods have potential pitfalls. PTW conducts a restricted search for a valid alignment that likely will not include the distortion between treatments or conditions. By contrast, DTW performs a relatively unrestricted search from a large family of possible alignments. (In practice, the search space is usually pared down by making certain assumptions about the warp, such as that it is global.) PTW is probably too limited, but we do not have enough data to take full advantage of the warping allowed by DTW. Is there a better way to align and compare gene-expression time-series data, that falls between the two methods? We have borrowed a technique from the chromatography community: the segment-based alignment. This is a piecewise linear alignment, that maps different segments of the time series separately. This allows adjacent time points to be warped in a similar manner. It permits a wider search than PTW, but not as wide as DTW. We describe two segment-based alignment methods we have developed in Chapter 5.

- Almost all previous work has made the assumption that the time series should be aligned globally. This is unwarranted for the tasks considered here. Keogh (2003) presents an approach that is the sole exception of which we know. Their method addresses this issue by separating series alignment into two steps. First this method finds a good ending time point for the query series, and then it performs a global alignment up to that point. Is there a way to perform non-global warping directly, as part of the alignment search? The alignment methods that we present and evaluate were developed to do precisely this. They align and short in a single step, performing a local alignment without a preprocessing step. We focus on a specific kind of local alignment, in which the end(s) of one series or the other remains unaligned. We refer to this as *shorting* the alignment. We believe that algorithms with this ability will outperform ones that cannot. We explain shorting in more detail in Chapter 2, and discuss how we incorporate it into alignment methods in Chapter 5.

- Previous approaches have assumed that all genes should share the same alignment, such as is illustrated in Figure 1.3. It has been conjectured that aligning the genes separately is prone to error, in part because they are sampled so sparsely (Bar-Joseph et al., 2003). Assuming this error is random, aligning all the genes together will average out this error. Is it really the case that all genes should always be warped as a unit? We observe that in some applications, the relationship between two time series may be different for different groups of genes. For example, when comparing an organism in which a gene has been knocked out with a wildtype one, there might be a specific group of genes affected by the knockout. Aligning genes in independent groups could reveal them. We therefore present and evaluate an approach that calculates clustered alignments of time series. We have devised a method that identifies clusters of genes, each of which is aligned independently. We discuss our method in detail in Chapter 7.

1.5 Related Tasks

The methods we develop here for comparing time series have applications in other fields. Many kinds of data are analogous to the data we use, in which several channels of data vary over time (or some other dimension). Such other domains include:

- *Speech recognition:* Dynamic time warping was originally developed in the speech recognition community (Sakoe and Chiba, 1978). A speech recognition system contains a database of spoken words, and compares utterances from a user against these database entries to find the closest matches. The sounds made can be warped to be stretched and contracted in time, and raised or lowered in pitch to match different voices.

- *Sign lanugage recognition:* Kadous (1999) has gathered channel-based data for Australian sign language. Here, the channels of data are the positions of the fingers and hands. An unknown hand sign can be contrasted with a database of annotated, known signs. Signs may need to be sped up or slowed down between replicates, or the positions of the hands may be slightly different. Kadous used a feature-extraction based approach to the problem,

classifying signs based on metadata from the observations. Keogh and Pazzani (2000) used a DTW approach on the same data that is closer to our methods.

- *Robotics:* Schmill et al. (1999) used dynamic time warping in the field of robotics. Their robot measures its environment using sensors, which include sonars, velocity encoders, bump sensors, and various readings from a blob vision system. Each of these sensors is a source of a discrete channel of information. In order to formulate a plan of action, the robot compares its sensor readings with previously seen ones. It uses DTW for this purpose.

- *Chromatography:* Nielsen et al. (1998) developed COW, or correlation-optimized warping, in order to process chromatograms. In this case, the channel of information is the output of a chromatography experiment, and the model lines up peaks from successive experiments in order to compare them. COW represents one of the first uses of a segment-based warping method. We discuss this approach more in Chapter 2.

1.6 Dissertation Statement

This thesis aims to answer several questions with regard to aligning and classifying gene-expression time series. Specifically, we focus on the following hypotheses:

i. Using higher-ordered interpolations when reconstructing unobserved data will result in more accurate alignments and comparisons.

ii. The best alignment methods are those which search a large space of potential alignments, yet are biased so that the warping function used is locally self-similar.

iii. Alignment methods that allow non-global alignments will result in more accurate aligments of related treatments.

iv. Warping all genes together will yield better results than warping each one independently, but grouping them into a small number of clusters to be warped separately may provide more accurate alignments.

1.7 Dissertation Outline

The remainder of this thesis is organized as follows:

- Chapter 2 delves into the background information that is necessary for this thesis. We first describe several biological concepts such as genes and nucleic acids, and the microarrays used to gather gene-expression data. We then discuss previous time-series alignment methods, and their inherent advantages and disadvantages.

- Chapter 3 describes some of the past work that has been done on related problems.

- Chapter 4 details the methods we have used in order to reconstruct unobserved data, interpolating these unobserved times within time series. We focus on the use of B-splines.

- Chapter 5 explains the segment-based methods we have developed in order to align time series and perform query similarity searches. We also discuss the methodology we have developed to assess the accuracy of similarity searches, when there are no exact matches to the queries within the database of treatments.

- Chapter 6 introduces optimization heuristics that we use in order to speed up our alignment computations without adversely affecting the accuracy of the resulting alignments.

- Chapter 7 discusses clustered alignments. A clustered alignment specifies sets of genes, each of which is aligned separately. We have developed a method that simultaneously decides to which cluster any one gene should belong, and aligns the genes within each cluster.

- Chapter 8 summarizes the key contributions of this work, discusses some open problems within time-series analysis, and offers some concluding remarks.

Chapter 2

Background

This chapter details background information necessary to understand the the remainder of this thesis. We start with basic biological concepts, including the nature of gene expression and how to measure it. We then detail previous methods that have been used to align time series.

2.1 Relevant Biology

Here we discuss the biological concepts necessary to understand this work, including genes, nucleic acids, the Central Dogma of molecular biology, and microarrays.

2.1.1 Genes and Gene Expression

Nucleic acids such as *ribonucleic acid* (RNA) and *deoxyribonucleic acid* (DNA) are used by living things to store and transmit the information necessary to sustain life. In bacteria and higher organisms, the primary "blueprint" is contained in DNA. A *gene* is a stretch of DNA that is the basic unit of heredity (Weaver, 2002). Most genes contain the information to create a *polypeptide*, which is a chain of amino acids. (Some genes do not encode polypeptides, while some make multiple ones.) A *protein* is one or more polypeptide chains that have been folded into a single structure. Proteins are the basic building blocks of cells, and include enzymes, hormones, structural elements, and molecules with many other functions.

Usually information cannot be transmitted directly from DNA to protein. There is an intermediate step: the *messenger RNA* (mRNA). mRNA is so called because its primary function is to allow the transfer of information from the DNA to proteins. This flow of information—DNA to

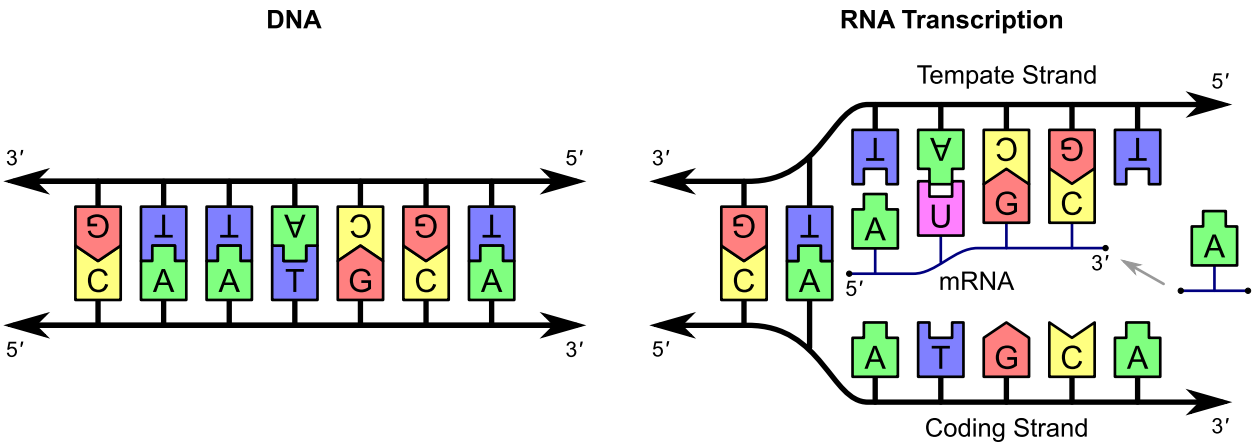

Figure 2.1: DNA and RNA transcription. The left panel shows a small stretch of DNA, with its two strands and "rungs" of base pairs. The right panel shows RNA transcription, in which part of the DNA strands separate and one of them is used as the template for synthesizing a new RNA molecule.

RNA, and RNA to protein—is called by the Central Dogma of molecular biology. (The Central Dogma also allows information to be transmitted from DNA to DNA, as when it is replicated.) There are some exceptions to this ordering (e.g. retroviruses and reverse transcription), but the vast majority of cellular activity follows this model.

A small strech of DNA is illustrated in the left panel of Figure 2.1. DNA consists of a sugar backbone of arbitrary length, connecting regularly spaced elements called *bases*. The backbone is directed, running from the "upstream" $5'$ (five prime) end to the "downstream" $3'$ (three prime) end. In the simplest model there are four bases: the purines *adenine* and *guanine* (commonly abbreviated A and G) and the pyrimidines *cytosine* and *thymine* (C and T). Almost all DNA occurs as the familiar double helix—two strands wrapped around each other, connected by "rungs" made of pairs of bases. The two strands are antiparallel, so that the $5'$ end of one strand coincides with the $3'$ end of the other. Usually, A and T will only pair with each other, and likewise for C and G. This means that the double helix is redundant, with each strand containing the complementary information content as its mate. This is crucial to DNA replication, in which the strands separate and each is used as the template for a new complementary strand.

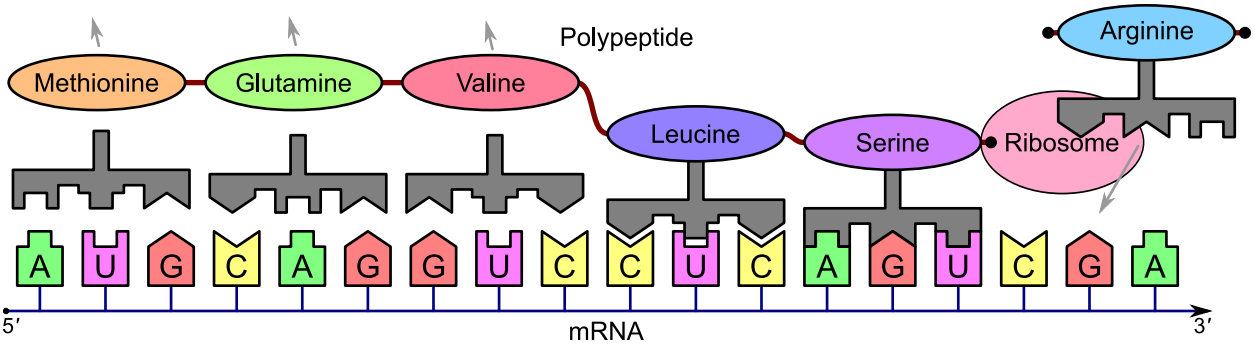

Figure 2.2: Protein translation. The mRNA molecule is used as a template for the new polypeptide. The tRNA molecule attached to each amino acid and the ribosome work in concert to map three base pairs to each amino acid.

RNA is very similar to DNA in structure. However unlike DNA it is typically found in single strands rather than in the double helix. In addition, the pyrimidine *uracil* (abbreviated U) takes the place of thymine. In *transcription* (shown in Panel B of Figure 2.1), a gene is copied from the DNA into a new RNA molecule. First the two strands of DNA separate from each other. One strand, called the *template strand*, is used to place new bases in the growing RNA. Complexes called RNA polymerases find the complementary base pairs and add them one at a time to the $3'$ end of the increasing chain of RNA, until the signal is received to stop. Thus the sequence in the nascent RNA strand matches the *coding strand* of the DNA, which is the strand not being used as the template.

Translation is the process of creating a new protein from a transcribed messenger RNA. It is illustrated in Figure 2.2. Like nucleic acids, proteins are composed of a chain of chemically similar elements. In the case of proteins, these elements are the twenty amino acids. It takes a set of three consecutive bases in the mRNA to map to a single amino acid. Such a set that is translated to an amino acid is called a *codon*. Each of the sixty-four (4^3) possible codons maps to either one of the amino acids, or serves as a signal that the chain is complete. The proper amino acid is placed in a growing chain with the help of *transfer RNA* (tRNA) molecules, which link the codons to the corresponding amino acids, and *ribosomes*, which facilitate the linking. When the process is

complete, the polypeptide will be folded into a chemical active protein (possibly along with other polypeptides), and released to perform whatever task it was created to do.

When a gene is *expressed*, it means that it is actively being used to create RNA. Most of these are mRNAs, but in some cases (like tRNA) the RNA molecule is the final product. A cell's genome may consist of hundreds to tens of thousands of individual genes. In multicellular organisms almost every cell contains a complete genome. In no cell is this genome uniformly expressed. The majority of a cell's energy is used in the creation of proteins, and fabricating them needlessly can be wasteful or even harmful. Some proteins should only be created under some special set of environmental circumstances, such as when some outside chemical has been introduced. Others may be crucial to a certain type of cell in a multicellular organism, but not needed in others. The system that regulates the expression of each gene is extremely complex. Concentrations of specific proteins can affect the rates of transcription of different genes. Small RNA molecules such as microRNAs (miRNAs) can also affect translation by regulating the activity of mRNAs. The machinery consists of a sequence of interrelated events in a "biological circuit." For example, a stimulus may cause a gene to be overexpressed, causing an abundance of its protein. This protein might block the transcription of another gene while increasing the transcription of a third, causing the appropriate effects on their proteins. This combination could in turn further increase the transcription of the first gene. Feedback loops, autoregulation, and circuit forks are common. With so many genes and their proteins present in each cell, deducing the sequence of responses to a stimulus and their effects can be difficult.

However, most regulation of protein synthesis takes place at the transcription level. Further, cells are continuously degrading mRNA after it has been transcribed. Thus the concentration of a certain mRNA is a good surrogate for the current activity level of a gene. Current technologies take advantage of this, by measuring the amount of different mRNAs in order to assess the expression of a gene.

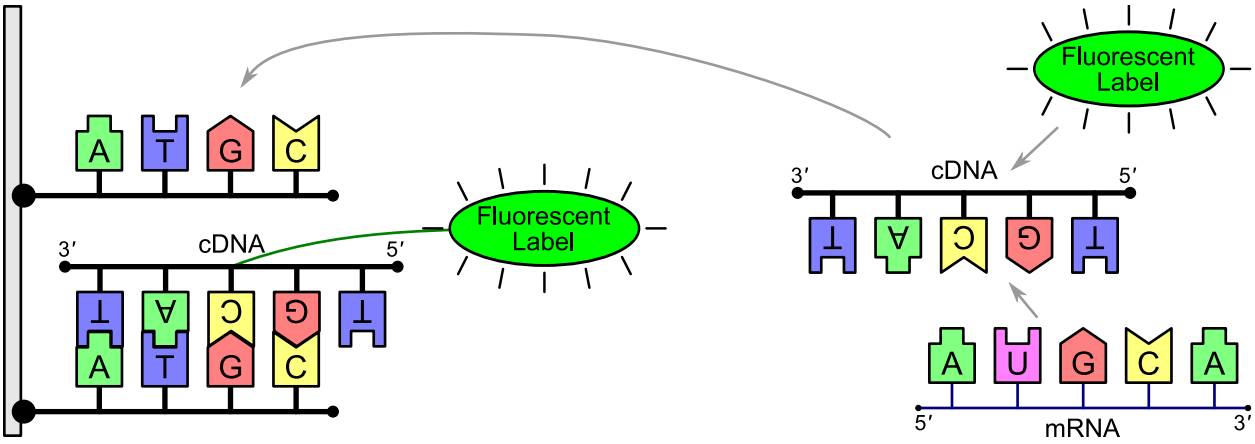

Figure 2.3: Microarray schematic. Complementary DNA or RNA (cDNA or cRNA) molecules are cloned from mRNAs and attached to a fluorescent label. These cDNAs are then exposed to the microarray. Each spot on it contains many repeats of the same sequence, attached to the glass or plastic base. If the labeled cDNAs find a spot with a complementary sequence, they will bond. Measuring the fluorescence will then reveal how much of the particular species of mRNA is present.

2.1.2 Microarrays

We have explained that the synthesis of proteins is vital to the continuing existence of the cell, and that the information necessary to construct them is contained within genes in the cell's DNA. Measuring the amounts of specific species of RNA present in a sample of cells gives a very good indication as to which genes are active.

Microarray technology (Schena et al., 1995) is a tool for measuring the relative concentrations of thousands of mRNAs simultaneously. At present, the use of microarrays is the most common method used for this task, even though they are often criticized for introducing large amounts of noise into the measurements (Aris et al., 2004). More accurate methods, such as RNA-seq (Wang et al., 2008) and Serial Analysis of Gene Expression (Velculescu et al., 1995)—which sequence individual RNAs—are becoming more common. As they become cheaper, microarrays will eventually be replaced by these technologies and the measurements made will be more reliable.

A microarray is a piece of glass or plastic that is covered with many small spots, each of which is keyed to a single sequence of DNA. Each spot contains several copies of a nucleotide

Table 2.1: Sample microarray data. TCDD refers to 2,3,7,8-tetrachlorodibenzo-*p*-dioxin. The expression values have been divided by control values and had their logarithms taken, so that positive values indicate increased expression, and negative values decreased expression relative to some baseline condition.

Treatment	Time	Dosage	Gene 1	Gene 2	Gene 3	Gene 4	...	Gene 1600
Acetominophen	4 hr.	100 $^{\mu g}/_{kg}$	-0.01	-0.01	-0.15	0.08	...	0.04
Acetominophen	12 hr.	100 $^{\mu g}/_{kg}$	-0.06	-0.11	0.15	0.26	...	-0.38
Acetominophen	24 hr.	100 $^{\mu g}/_{kg}$	-0.16	0.19	-0.74	0.42	...	-0.03
Acetominophen	4 hr.	500 $^{\mu g}/_{kg}$	0.08	0.21	-0.36	0.10	...	0.18
Acetominophen	12 hr.	500 $^{\mu g}/_{kg}$	-0.06	-0.03	0.08	0.10	...	-0.03
Acetominophen	24 hr.	500 $^{\mu g}/_{kg}$	-0.03	0.28	-0.47	0.59	...	0.32
TCDD	6 hr.	1 $^{\mu g}/_{kg}$	-0.01	-0.10	-0.14	0.28	...	0.03
TCDD	12 hr.	1 $^{\mu g}/_{kg}$	-0.06	0.01	0.04	-0.11	...	-0.12
TCDD	24 hr.	1 $^{\mu g}/_{kg}$	-0.06	0.01	0.08	0.10	...	0.00
TCDD	36 hr.	1 $^{\mu g}/_{kg}$	0.16	0.03	0.01	0.26	...	-0.04
TCDD	48 hr.	1 $^{\mu g}/_{kg}$	-0.03	-0.19	0.12	-0.04	...	0.01
TCDD	60 hr.	1 $^{\mu g}/_{kg}$	-0.08	0.04	-0.07	0.43	...	-0.03
TCDD	72 hr.	1 $^{\mu g}/_{kg}$	-0.03	-0.03	-0.08	-0.07	...	0.01
TCDD	96 hr.	1 $^{\mu g}/_{kg}$	-0.11	-0.16	-0.08	0.41	...	-0.06

sequence, which are bonded to the chip as shown in Figure 2.3. Each sequence will bond to a free-floating nucleotide strand with a basewise complementary section. These strands are made by cloning them from the mRNA present in a given biological sample. In some systems, these strands are complementary DNA (cDNA), which is created with reverse transcriptase (which is an enzyme capable of synthesizing DNA from RNA). Other systems use complementary RNA (cRNA), which can be created from the cDNA. The number of complementary strands (whether cDNA or cRNA) is proportional to the original number of mRNAs. Further, each one has had a fluorescent molecule attached to it. It will then *hybridize*, or bond, to a spot on the microarray only if some part of its sequence complements the one on the spot, attaching the label to the array. Thus, the strength of the fluorescence of a spot will provide a signal proportional to the concentration of the original mRNA, and indirectly the expression level of the associated gene.

Table 2.1 shows an example of microarray data, taken from the EDGE project (Hayes et al., 2005). Here each expression value is the average liver cell expression for four different mice that

have been exposed to the given treatment, divided by the average expression of four control mice. Then the base-e logarithm is taken, so that a positive value indicates increased expression and negative values indicate decreased expression. This is the type of data that we use to create our time series.

As stated before, microarray measurements are subject to noise (Aris et al., 2004). Such noise includes:

- *Nonspecific cross hybridization.* The nucleotide sequences connected to the microarray may in fact match subsequences of many genes, not just one. This can result in observations in which one gene is counted as another.

- *Differences in the efficiency of labeling reactions.* Different nucleotide sequences may be more or less likely to bond with the fluorescent label. Also, sometimes the label itself may alter the degree to which the nucleotide can bond to the array.

- *Production differences between microarrays.* The production process is not perfect, and differences between microarrays can cause small differences between observations.

As microarrays are replaced by newer technology, this noise will be minimized. However even with the most accurate of methods, biological variability remains a complicating factor. Different observations are often of different organisms, who respond in subtly different ways to stimuli. This is especially the case when working with mice or other animals, in which an observation often requires the sacrifice of the animal.

In addition, each observation (regardless of method) can cost hundreds or thousands of dollars. This means that time-series data often contains few observed times, and few replicates. This temporal sparsity is another potential source of error, as it is difficult to correct noise that has been introduced.

2.2 Alignment of Time-Series Data

In some cases, a gene-expression response may be assayed at various time points, and thus the measured data constitutes a time series. One task of interest might be to compare pairs of

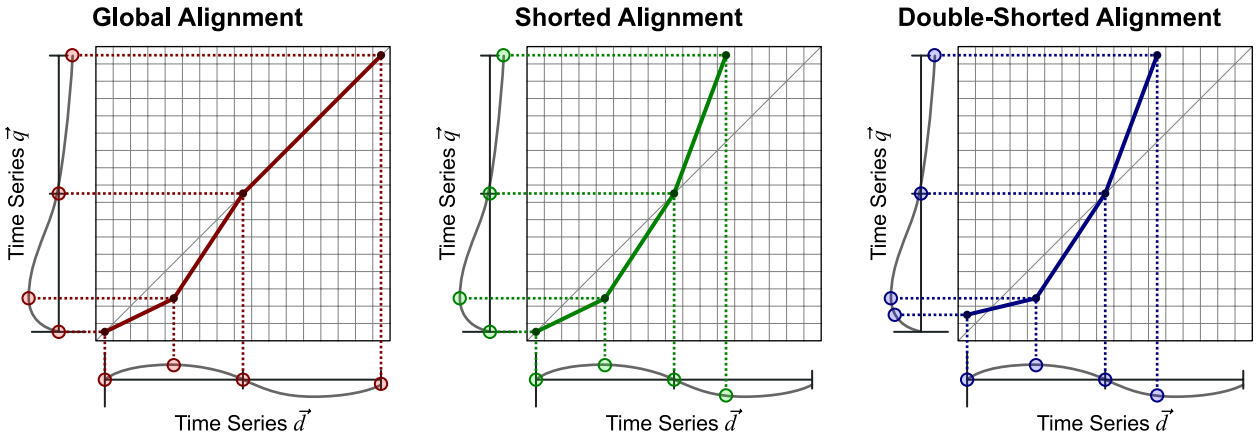

Figure 2.4: Global and shorted alignments in warp space. The alignment paths show the mapping between two time series. Corresponding points are highlighted. In the global warping, the end points are aligned. In the shorted alignment, the end of one series is not aligned to the other. In the double-shorted alignment, the end of either series and the beginning of either series are not so aligned.

time series, in order to assess their similarities and differences. However, one cannot usually just compare identical times within the two series (i.e. comparing the series at zero hours, then at one hour, then two hours, etc.). This has been shown to be a very brittle measure that is especially vulnerable to temporal shifts or noise in the series (Keogh and Pazzani, 2000). Instead, one must first *align* the series in order to determine equivalent times.

The alignment of two time series can be easily visualized in *warp space*. Warp space is a two-dimensional Cartesian coordinate system, in which both dimensions are time. One dimension is the time of the first series, and the other is the time of the second series. Examples of warp space can be seen in Figures 2.4 through 2.7. We call the first series $\vec{d}$ (for database) and the second series $\vec{q}$ (for query).

An *alignment path* or *warping path* is a continuous set of points (x, y) in warp space. They are restricted so that the values in each dimension are monotonically increasing. That is,

$$x_i \leq x_{i+1} \tag{2.1}$$

and

$$y_i \leq y_{i+1}. \tag{2.2}$$

The alignment path corresponds to a mapping from one series to the other. If a path contains a point (x, y), then time x in $\vec{d}$ maps to y in $\vec{q}$. Small horizontal or vertical sections of the path are analogous to "gaps" in sequence alignment (Durbin et al., 1998). If there are no such horizontal or vertical sections, the mapping between the series will be one-to-one. Figure 2.4 illustrates three different warping paths. Intuitively, one can think of the alignment path as "starting" at its lowermost, leftmost point and proceeding up and to the right. For any point on the path, one can trace a horizontal line to the left and a vertical line down, to find the corresponding points in the two series. Several of these correspondences are shown in the figure.

Note that the methods detailed here can be used to align either a one-channel time series, such as the expression profile of a single gene, or a multi-channel time series, such as the expression profile of a set of genes. For clarity, the figures show a single gene, even though that is seldom the case.

There are several methods used to search for the path which best aligns the two series. We will explore several of them in this chapter, including parametric time warping, dynamic time warping, and correlation-optimized warping.

2.2.1 Alignment Shorting

The vast majority of work on time-series alignment assumes a global alignment, in which the whole of one series is aligned with the whole of the other (Keogh and Pazzani, 2000; Aach and Church, 2001; Bar-Joseph et al., 2003; Ratanamahatana and Keogh, 2005). This is the kind of alignment that is shown in the first panel of Figure 2.4. By definition, a global warp assumes that the first times of the time series correspond to each other, and the last times as well. In warp space, this means that a global alignment path will stretch from the origin (bottom left in the figures) all the way to the farthest point (top right).

This assumption is often unwarranted in gene-expression time-series problems. For example, consider the case in which we are comparing two related treatments, in which one has a greater dosage or efficacy. In such a case, its effects might appear greatly accelerated when compared to the other. The algorithm should not align the entirety of both series to one another. Rather, it

should align all of the weaker treatment's series to the first part of the stronger treatment's series, leaving the rest of the weaker treatment unaligned.

We call such an alignment a *shorted* alignment, and show an example in the center panel of Figure 2.4. A shorted alignment is a special case of a local alignment. We still require the beginnings of both series to be aligned to each other. However the end of one and only one series can remain unaligned. In alignment space depictions like those in Figure 2.4, a shorted alignment path begins in the lower lefthand corner, and stretches to either the top or to the right of the graph. Shorted alignments are most useful when treatments have well-defined zero-points that one can assume are aligned. For example, in the toxicogenomic studies we consider, the zero-point is the time at which the toxin was introduced to the organism. Here we can make the simplifying assumption that before the treatment is applied, all the animals were expressing their genes in a similar manner.

Note that we are not the first group to recognize the need to develop algorithms for computing shorted alignments. Keogh (2003) devised a two-step shorting method that first finds the appropriate end points of an alignment, before calculating a "global" alignment up to these points.

As we show in the next sections, different search algorithms for the best alignment path are able to short to differing degrees. One of our hypotheses is that shorting is an important consideration in aligning time series, and methods that are able to do this will tend to perform better than similar methods that cannot.

In some cases, we may wish to allow the alignment to short not only at the end of the series, but also at the beginning. Such a *double-shorted* alignment is shown in the rightmost panel of Figure 2.4. We may wish to use this type of alignment in problems in which there is no well-defined zero-point in the time series. For example, when the "treatments" consist of organisms with different genotypes, there may not be times that we can assume a priori are aligned to each other. In a double-shorted alignment, the beginning of one of the series remains unaligned, and the end of one of the series (possibly the same one) also remains unaligned. In terms of the warping paths in our figures, a double-shorted path will start on the bottom or the left, and extend to the top or to the right.

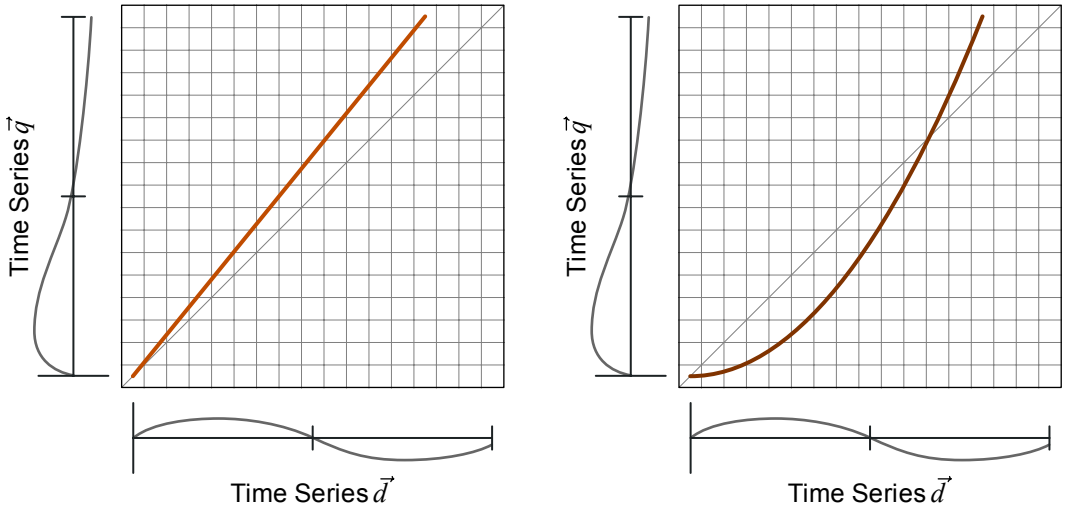

Figure 2.5: Parametric time warping. The alignment is selected from a specified family of functions, such as the set of linear or second-order polynomial alignments exemplified here. Finding the alignment consists of fitting the parameters of the function in order to minimize a distance function.

Unfortunately, determining a double-shorted alignment is often computationally much more expensive than a single-shorted alignment or a global alignment. Many of the path search algorithms rely on dynamic programming, and allowing both variable start points and variable end points limits the utility of storing calculations in a table or matrix. Further, one must take care that the warping path aligns a significant portion of both series. For example, an algorithm might align the very beginning of one series with the very end of the other. This would result in a very short alignment path, in one of the corners in the warp space figure. While technically a proper double-shorted alignment, this will often not be a very interesting result.

2.2.2 Parametric Time Warping

Parametric time warping (PTW) is a simple warping method, which was formalized by Eilers (2004). Parametric warping includes linear warping, which has often been used before (e.g. by Bar-Joseph et al. (2003)). The idea of PTW is to use a function $P(t)$ to characterize the warping path, where P is chosen from a predetermined family of functions $\tilde{P}$ (e.g. polynomials of a certain order). Ideally, we can find a function such that:

$$d_{P(t)} \approx q_t, \tag{2.3}$$

where d_i and q_i are the ith elements of $\vec{d}$ and $\vec{q}$. Most commonly, this means minimizing the squared distance between the two series:

$$P = \underset{\tilde{P}}{\operatorname{argmin}} \left(\frac{1}{|\vec{T}|} \sum_{t \in \vec{T}} D(q_t, d_{\tilde{P}(t)}) \right), \tag{2.4}$$

where D is a distance function between two points q_t and $d_{\tilde{P}(t)}$ in the time series being compared. Usually, D is squared Euclidean distance. The vector $\vec{T}$ is the set of times for which the distance will be calculated, such as an evenly spaced distribution over $\vec{d}$. Because Equation 2.4 is not a linear equation, there is no analytic solution. It must be solved via a search or by successive approximations. In principle, the set $\tilde{P}$ may be any valid family of functions. In practice, it is usually limited to polynomials. For many applications, second-order polynomials are sufficient (Eilers, 2004):

$$P(t) = a_0 t^2 + a_1 t + a_2. \tag{2.5}$$

The values of the coefficients are solved via successive approximations until Equation 2.4 has been minimized.

Parametric time warping is fast, is easy to implement, and can return accurate alignments when the proper family of functions is known. Further, no special care needs to be taken to allow the method to find shorted alignments. If the function naturally increases in one dimension faster than the other, the alignment will be shorted.

However, PTW can have considerable disadvantages. Most importantly, the search is limited to a single family of functions. Many possible alignments are not even considered. In statistical terms, the resulting alignments may have a large bias component to their errors (Geman et al., 1992). Another potential problem is that the distances between any two points d_i and q_j in the time series are usually calculated independently of each other. This means that the algorithm does not account for stretches of the time series in which one of them undergoes some uniform shift.

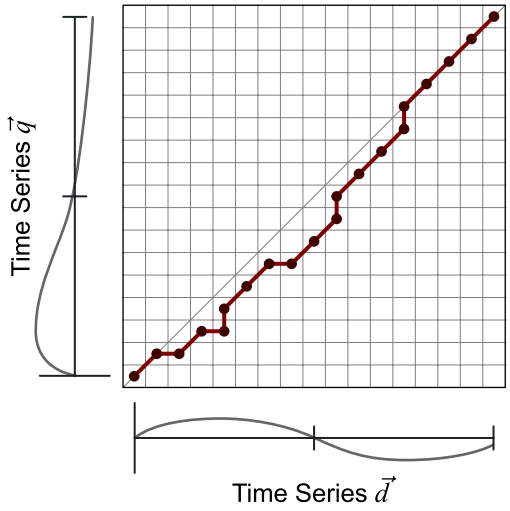

Figure 2.6: Dynamic time warping. Individual points of the series are compared via Euclidean distance, creating a warping path with many short vertical, horizontal, and diagonal segments.

For example, if the two series are related but one of them has a higher dosage or efficacy, there might be a uniform amplitude shift in a stretch of that series. In theory, redefining D to remove this independence assumption could help compensate for this. However in practice, this would drastically increase the search time necessary, and to our knowledge has not been done.

2.2.3 Dynamic Time Warping

Dynamic time warping, (DTW) is one of the most common methods used to align two time series. It was developed by Sakoe and Chiba in the speech processing community (1978), but has been applied to a variety of domains including robotics (Schmill et al., 1999), hand gesture recognition (Keogh and Pazzani, 2000), manufacturing (Gollmer and Posten, 1995), and medicine (Caiani et al., 1998). It is a dynamic programming algorithm, taking advantage of frequent repeated calculations.

Figure 2.6 shows an alignment returned by DTW. It computes the alignment of series $\vec{q}$ and $\vec{d}$ by creating an *alignment matrix* $\mathbf{\Gamma}$. Each element $\gamma(i, j)$ holds the score of the best alignment of $\vec{q}$ up to time i against $\vec{d}$ up to time j. The elements are calculated recursively as:

$$\gamma(i,j) = D(q_i, d_j) + \min \begin{cases} \gamma(i-1,j) \\ \gamma(i,j-1) \\ \gamma(i-1,j-1) \end{cases} . \tag{2.6}$$

Here, $D(q_i, d_j)$ represents the Euclidean distance between between $\vec{q}$ at point i and $\vec{d}$ at point j. The first element of $\mathbf{\Gamma}$ is defined as:

$$\gamma(0,0) = D(q_0, d_0). \tag{2.7}$$

The final element, $\gamma(|q|-1, |d|-1)$, thus holds the score for the best alignment of the entirety of both series. This is used as the final score:

$$\text{score} = \gamma(|\vec{q}|-1, |\vec{d}|-1). \tag{2.8}$$

Starting at this element, we find the previous adjacent best element that was used to define that element's score, as per Equation 2.6. We continue *tracing back* until we reach $\gamma(0,0)$. These elements found by the traceback define the alignment path. Formulated in this way, DTW is incapable of shorting an alignment. (Though as mentioned previously, Ratanamahatana and Keogh (2005) short by first taking an extra step to search for subseries to "globally" align.) In Chapter 5 we will discuss a method we have developed to allow DTW to short alignments, by redefining Equation 2.8.

Given series of length m and n, the alignment matrix has mn entries to be calculated. Each of these calculations takes constant time. Thus, if $m \approx n$, the time and space complexities are $O(n^2)$. Keogh (2002) has been able to speed up alignments even further, to approach $O(n)$. He does this by simultaneously restricting the allowed warping path to be close to the diagonal, and quickly calculating lower bounds on their distance functions that allow many of the calculations to be skipped.

The alignment space searched by DTW is quite large. With a fine enough temporal granularity in the series being aligned, the warping path returned can arbitrarily approach any path that monotonically increases in both dimensions from the origin to the end point. However, in practice the

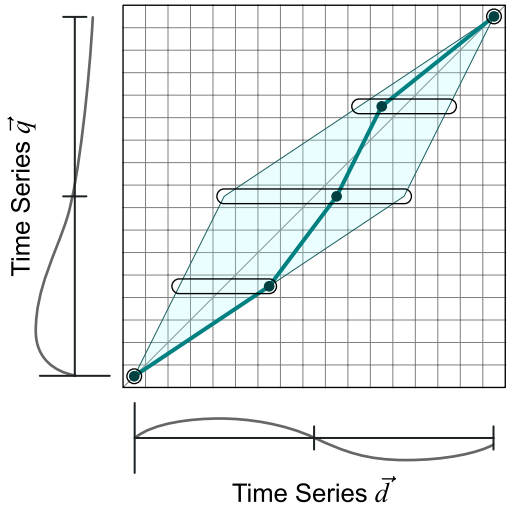

Figure 2.7: Correlation-optimized warping. COW is a segment-based warping method that searches for the best knots (points of discontinuity of the segments) only within the marked tracks.

method shows a strong bias toward returning a short path. This is because almost every step adds to the total score, which the algorithm is minimizing.

Like parametric time warping, DTW also makes an independence assumption that can cause problems. Because of the nature of the warping matrix, the distance between any pair of points q_i and d_j is necessarily independent of the distances between other points. If there is a stretch of several consecutive times in one of the series that undergoes the same shift (such as a uniform amplitude shift), DTW will not take that into account. It will score each individual time pair separately.

Nevertheless, DTW remains in common usage in many domains. It is fast, well-studied, and fairly accurate.

2.2.4 Correlation-Optimized Warping

Correlation-optimized warping (COW) was developed by Nielsen et al. (1998) in order to compare chromatography data. As shown in Figure 2.7, COW is a segment-based method. The alignment path consists of m contiguous segments. These segments partition the series $\vec{q}$ and $\vec{d}$ each

into m subseries, where the ith subseries of each series correspond to each other. The score of a given alignment is the sum of the correlations between corresponding subseries.

Figure 2.7 also illustrates how COW searches for good segment endpoints, of *knots*. It looks for these in only a limited area of alignment space. The segments are assumed to be of constant length with respect to $\vec{q}$, and variable with respect to $\vec{d}$. The vector K contains the coordinates of the knots in $\vec{q}$. These are usually evenly spaced. COW works by filling a zero-indexed matrix $\mathbf{\Gamma}$, which is of dimensions $m+1$ by $|d|$. The element $\gamma(k,i)$ contains the score of the best alignment of $\vec{d}$ from zero to i and $\vec{q}$ from zero to K_k (the kth element of K) using k segments. It is filled using the following recurrence relations:

$$\gamma(0,i) = \begin{cases} 0 & \textbf{if } i = 0 \\ -\infty & \textbf{otherwise} \end{cases} \tag{2.9}$$

$$\gamma(k,i) = \max_{y \in \mathrm{pred}(i,k)} \left[\gamma(k-1,y) + \mathrm{cor}\left(d(y,i), q(K_{k-1}, K_k)\right) \right] \tag{2.10}$$

where $q(t_0,t_1)$ represents a subseries of $\vec{q}$ from t_0 to t_1, $d(t_0,t_1)$ is defined likewise, and $\mathrm{cor}(\vec{a},\vec{b})$ is the Pearson correlation (Coolidge, 2006):

$$\mathrm{cor}(\vec{a},\vec{b}) = \frac{\sum_{i=0}^{|\vec{a}|-1}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{(\sum_{i=0}^{|\vec{a}|-1}(a_i-\bar{a})^2)(\sum_{i=0}^{|\vec{b}|-1}(b_i-\bar{b})^2)}} \tag{2.11}$$

The predecessor function lists valid starting locations in $\vec{d}$ for segments ending at i:

$$\mathrm{pred}(i,k) = i - \frac{|\vec{q}|}{|\vec{d}|}(K_k - K_{k-1}) - t, ..., i - \frac{|\vec{q}|}{|\vec{d}|}(K_k - K_{k-1}) + t, \tag{2.12}$$

with t being a user-defined "slack parameter" that controls the size of the search space.

The best alignment, and its resulting score, is represented by the element of γ that corresponds to the end of the global alignment:

$$\mathrm{bestscore} = \gamma(m, |\vec{d}| - 1), \tag{2.13}$$

Assuming the segments are of roughly equal length, and that the slack parameter is proportional to this length, COW's time complexity is $O(mn^3)$, where m is the number of segments and n is

the length of one of the series. Each segment calculation can be done in $O(n)$ time. The number of predecessors for any one element of one of the matrix is also $O(n)$. The size of the matrix (and the space complexity of the algorithm) is $O(mn)$. Multiplying these values together yields $O(mn^3)$.

Note that COW does not make the independence assumption between points in the series that both DTW and PTW do. Within the subseries defined by each segment, the distances between points are clearly dependent on each other.

However, COW has several limitations. First, it is incapable of shorting the alignment. Because it only searches for knots in one dimension, it is difficult to modify the algorithm to allow shorted alignments. Second, COW is apt to align subseries which differ greatly in magnitude because it scores by correlation. Third, the computation in Equation 2.10 may sometimes return an undefined value if the input segments do not have a defined correlation (as when both segments consist of all zeros).

2.3 Chapter Summary

In this chapter we have provided reviews of the necessary background knowledge to understand this dissertation. First, we covered relevant biological concepts, including DNA, RNA, and proteins, the Central Dogma of molecular biology, and microarrays. We described the methods used to obtain the data with which we will be working. Second, we explained previous algorithms used to align time-series data. Parametric time warping finds a function from a family that best aligns the series. Dynamic time warping uses a fast dynamic programming algorithm to create a path from many short sections. Correlation-optimized warping is a segment-based method that scores every segment via correlation.

Chapter 3

Related Work

In this chapter we describe some of the previous, related work that other groups have done. We start by describing methods that explicitly align pairs of time series. Techniques in this group tend to be the most related to our own work. After this, we review other works that are related to our own, including techniques to model time-series data (such as dynamic Bayes networks), and methods that adress problems similar to specific subtasks of our alignment problem (such as reconstructing hidden expression level data, or clustering time series).

3.1 Time-Series Alignment Methods

We start by detailing works whose primary mechanism of action involves aligning two or more time series. Often this is the first step toward assessing the similarity of the time series, and this assessment may in turn be used to find the most similar series to a query.

3.1.1 Dynamic Time Warping

Much of the work in alignment has focused on dynamic time warping (DTW), which was originally developed by Sakoe and Chiba (1978) in the speech recognition community. Aach and Church (2001) were the first to apply the method to gene expression profiles, and other groups have followed (Liu and Müller, 2003; Criel and Tsiporkova, 2006). The alignment methods that we present in Chapter 5 differ in several key respects. Most significantly, we use a segment-based alignment method rather than DTW. In addition, Aach and Church focused only on alignment, while our methods also are able to pick out the database time series most similar to a query series.

Further, we use nonlinear spline models in conjunction with time warping in order to interpolate to unseen time points, while Aach and Church used only linear interpolation. Finally, we consider local alignments of time series in which one of the series is shorted.

Keogh and Pazzani (2000) explored using DTW to compare a query to each series in a large database in order to classify the query. They observed that the time complexity of DTW, $O(n^2)$, can be too large when comparing a query against a large number of other series. Their solution was "piecewise aggregate approximation," in which they subsampled the series being compared and calculated the distance between the subsampled series as a first approximation of the true distance. Doing so does not significantly affect accuracy of classification, although it increases the speed of the calculation by one or two orders of magnitude.

Keogh (2002) continued to improve DTW's running time by restricting the allowed warping path to be close to the diagonal—keeping allowed warps within the so-called "Sakoe-Chiba band." With most warps disallowed, a given point in one series is compared to only a limited number of points in the other series. It is easy to calculate the range of values those other points might have, thus being able to quickly find a lower bound on the distance to be calculated. This speeds the alignment calculation substantially. On 32 different data sets, Keogh showed the complexity to be essentially $O(n)$.

Keogh (2003) continued using this bounding approach to DTW. In this work, it was applied to "uniform scaling," in which one series is stretched or compressed by a constant factor before aligning. This approach is very close to the "shorting" that we have studied, and predates our work. Fu et al. (2008) synthesized the bounding functions for uniform scaling and DTW into a single bound, while Euachongprasit and Ratanamahatana (2008) added amplitude shifting to the method.

Ratanamahatana and Keogh (2005) dispelled some common misconceptions about DTW, and made compelling arguments in light of these studies for using DTW for time-series alignment problems. They argued that DTW is simple, fast, and accurate, and should be considered for any time-series alignment problem. However in our case, DTW does not have as high an accuracy as the segment-based alignment methods that we use. We believe there are a few reasons for this.

First, DTW's search for an alignment path has few restrictions, and it can return an approximation of any valid alignment function (see Section 2.2). The Keogh group's use of the Sakoe-Chiba band in conjunction with uniform scaling addresses this problem, but creates another: it assumes that the warping of one series onto another should be roughly uniform. Gene-expression time-series data can violate this assumption—especially in cases in which the two series being aligned are from similar but not identical treatments, or when they are from treatments with different doses. Secondly, DTW has no memory of previous calculations. Every point-to-point distance is necessarily independent of distances immediately before and after it. This is the one reason that we use a segment-based alignment. Within each segment, each time point is warped in the same way as its neighbors, and distances of adjacent times are dependent on one another.

Finally, Xi et al. (2006) made contributions in the area of what they call "numerosity reduction" with respect to the classification task using DTW. They assume that the database holds many example series with the same classification. Their technique discards redundant or confusing examples, speeding up the search for ones similar to the query. This is not a valid strategy in our case because the members of our database are already composite objects formed from numerous discrete observations. We do not have replicate time series of the same treatment, as might be the case in domains besides gene expression.

3.1.2 Segment-Based Warping

Nielsen et al. (1998) developed COW, or correlation-optimized warping. COW is the first example of a segment-based alignment method, and was created to compare chromatography data. It divides the two series into m segments each, and then sums the Pearson cross correlations (Coolidge, 2006) of each corresponding segment, using dynamic programming to find the alignment that maximizes the sum. However it has some drawbacks. Most significantly it cannot short. Additionally, the use of correlation means that it is blind to large differences in amplitude between each segment pair. Despite these problems, our tests show it to have a high accuracy in finding similar gene-expression time series. We have developed our SCOW algorithm (see Section 5.1.2) in order to solve these shortcomings of COW.

3.1.3 Parameterized Time Warping

Eilers (2004) formalized the use of parametric time warping, in which the alignment path is selected from a family of closed-form functions (e.g. linear warping, polynomial warping, etc.). However the concept had been used many times before, as by Bar-Joseph et al. (2003). The principal problem with parametric time warping is that one must know the proper family of functions a priori. We use parametric time warping as one of our comparison methods, and find that it does not align or classify well when using our toxicology time-series data set.

Bar-Joseph et al. (2003) used a warping method that finds a linear mapping between the two time series being aligned. Although it allows local alignments like our method, the linear model does not adequately represent complex alignments. Our alignment methods are more expressive, being based on multisegment methods (although not as expressive as alignments based on DTW).

3.1.4 Latent Trace Models

Listgarten et al. (2005) developed a generative model that aligns multiple time series of the same kind or class. Their model assumes that each observed time series is a non-uniformly subsampled version of a single, unobserved "latent trace." Each observed series also has some local rescaling and additive noise. They did not use their method for similarity queries, but only to align multiple series together.

3.1.5 Longest Common Subseries

Another approach to aligning time series is the "longest common subseries" (LCSS) approach. In this approach, the similarity between two time series is the length of the longest common sequence between the two, often normalized by the length of one of the input series. Bollobás et al. (2001) modified the definition of LCSS by allowing one of the series to be transformed by a linear function. This could be particularly useful for gene-expression time-series data, since similarity functions need to be robust to amplitude changes. However Bollobás et al. did not apply their method to gene-expression analysis. The authors detailed several algorithms for calculating LCSS with linear transformation: an exact one that runs in cubic time with respect to the length of the

series, and three heuristics that run in linear to quadratic time. However, they did not address the issues of stretching series temporally, or of shorting the series.

Morse and Patel (2007) developed another algorithm for calculating the exact LCSS of two series. This one has a worst-case time complexity of $O(n^2 3^d)$ and a space complexity of $O(n^d)$, where n is the length of the series and d is the dimensionality of the data. However, they showed that it often works in much better time, in cases where d is low enough that it can be ignored. They achieve this speedup by placing every observed point in a d-dimensional grid, and pruning comparisons of points that are not close in the grid. Unfortunately, this approach suffers from the curse of dimensionality. Morse and Patel focused on data with only one or two dimensions. In gene-expression time-series data, every gene is a separate dimension. With thousands or tens of thousands of dimensions, the method cited here would have difficulty computing the distance between time series in a reasonable period of time. The authors suggest handling high-dimensional data by projecting it into a much lower-dimensional space, in order to place them on the grid. However with such an extreme change in dimensionality, it is unclear if this would offer any benefit over other alignment methods. Further, their method suffers from the same problem as DTW, in that every pair of times is compared without consideration for other adjacent time points.

3.2 Other Work

There has also been substantial research in other related areas besides time-series alignment. Many of the following methods are relevant to our work.

3.2.1 Signature-Based Identification

Thomas et al. (2001) and Natsoulis et al. (2005) were both concerned with finding "signatures" of gene expression levels for purposes of classification among treatments or conditions. However unlike our work, they did not use time series in their calculations. Instead, they treated each gene as a single scalar value, and attempted to classify a query based on these values. (For example, as exposure to a class of chemical, or as a particular disease.) Thomas et al. used a naïve Bayes approach, while Natsoulis et al. compared several different algorithms, including support vector

machines and linear classifiers based on logistic regression. Both also devoted considerable effort to reducing the number of genes used in the calculation, in order to make the algorithms more human-comprehensible.

Lamb et al. (2006) addressed a similar task, except that they took a nonparametric, rank-based approach. Using their method, one builds a database in which each element is an ordered list of the genes under consideration. The genes at the top of the list are those that are most strongly expressed (relative to a control) at a specified time after exposure to a treatment. Likewise, the genes at the end are those that were most weakly expressed. The query is then compared to each element in the database using a rank comparison method similar to Spearman's rank correlation coefficient, to obtain a score between -1 (meaning strong anticorrelation) and 1 (strong correlation).

3.2.2 Dynamic Bayes Networks

Ong et al. (2002) used dynamic Bayes networks (DBNs) to model the expression levels of genes within an organism over time. In this case the organism was *E. coli*, and they were trying to model the dependencies of its genes on one another. Their method makes extensive use of expert domain knowledge in order to create an initial DBN, and then uses standard techniques to modify it according to the data. Nachman et al. (2004) also used DBNs, but in conjunction with a kinematic model that explicitly models the relation between concentrations of unobserved regulatory proteins and gene expression levels.

3.2.3 Feature Extraction

Other methods work by extracting features from the temporal data in such a form that can be used by machine learning algorithms such as decision trees, neural networks, inductive logic programming, etc. (Mitchell, 1997). This is the approach taken by Kadous (1999) to compare a sign in Australian Sign Language against a database of known signs. In this case, the temporal data consists of the locations of the hands and fingers over time. Kadous's emphasis was on extracting human-meaningful rules to classify each sign.

Eruhimov et al. (2007) developed a similar algorithm, also extracting features from the time series. However the authors were not concerned with human comprehensibility. Instead, their algorithm extracts a large number of statistical measures characterizing each dimension, and then classifies time series via a boosted ensemble of trees. Such an approach radically increases the number of features, and may not be tractable for gene-expression time-series data, which can already contain many thousands of genes.

Liu and Müller (2003) used a model which calculates the mode of a series, which is a nonparametric summary statistic. Their technique couples mean updating with time warping in order to obtain the mode of a set of time-warped gene-expression series. The modes can then be used to characterize individual genes, or to group them.

Another example of this kind of work was Yamada et al. (2003). The technique described here couples DTW with decision trees (Mitchell, 1997), to make a human-comprehensible tree that classifies time series. In their scheme, every leaf node of the decision tree contains a classification, while every internal node of the decision tree consists of a time series and a threshold. Each internal node has two branches. If the distance between the query series and the node series is less than the threshold, the query goes down one branch. If not, it goes down the other. The decision tree is created by taking each series from the training set in turn, and judging how well it splits the data into different groups by adding an optimal threshold.

3.2.4 Reconstruction of Hidden Data

Given the sparseness of microarray data, the reconstruction of hidden time points has been an important question in its own right. Bar-Joseph et al. (2003) and Luan and Li (2004) were among the first to use B-splines to represent missing data. The methods we detail in Chapter 4 extend their work. Our techniques smooth the data, making our reconstructions less sensitive to outliers. We also make sure that the linear equations used to fit the splines are solvable, which is a possible pitfall in the works listed here.

Other authors have tried to use specific domain knowledge to model the prevalence of mRNA following a treatment. Chechik and Koller (2009) created an “impulse model” that specifically

models the starting equilibrium of specific mRNA molecules, a short period of transition after they have been perturbed, and a new, different equilibrium after the reaction has come to rest. This model is especially appropriate in domains such as toxicology, in which a specific treatment is introduced at a known time. Farina et al. (2008) provided another kinematic model, this time explicitly incorporating the rate of mRNA degradation over time.

The work of Nachman et al. (2004) is also related to the issue of data reconstruction. As stated above, they use a kinematic model which explicitly represents the relation between hidden regulator proteins and gene expression levels. They use this model to reconstruct the levels of different mRNA molecules in a sample at unseen times. They also use it to find which regulators affect which genes, by creating an "ideal regulator" that would explain all of a gene's unexplained activity, and finding which known regulators most closely match this profile.

3.2.5 Clustering of Time Series

Finally, there is a wealth of previous work on partitioning genes into meaningful clusters based on their expression levels. However our work on clustering is novel mainly in that we do not cluster directly on expression. Rather, we cluster based on the difference in expression between two different treatments (i.e. conditions). In our scheme, two genes with radically different expression profiles may be in the same cluster, if those profiles are altered in the same way between two treatments.

Ben-Dor et al. (1999) used a graph-theoretical method to cluster genes. Their method creates a graph in which every node is a gene. Every gene is connected in the graph to all the others with which it has a similar profile. It then finds all the cliques or near-cliques in the graph. The method does not explicitly use time in its calculation, although the authors suggested that it can be made to do so by measuring all genes at several predefined time points, and/or using differences between subsequent times as features.

Chudova et al. (2003) were concerned with simultaneously aligning and clustering sets of multidimensional curves, using an expectation-maximization (EM) algorithm. During each iteration of the EM procedure, the parameters of each cluster of genes are updated, and then the genes are

reclustered according to the new parameters. In this sense the approach is similar to our own clustering methods. However, they used a Markov model to perform time warping, in which each state represents a stage in the gene's evolution over time.

Gaffney and Smyth (2005) also used an EM procedure to simultaneously align and cluster genes. However, their method warps genes directly against each other, rather than using a Markov model. Their method uses a combination of linear warping along with regression mixture models that allow for nonlinear warping.

Yuan and Kendziorski (2006a) used hidden Markov models (HMMs) to efficiently identify differentially expressed genes in gene-expression time-series data. The data their method analyzes must be taken from at least two treatments or conditions, and the states of the HMM indicate whether the gene is expressed equivalently or differentially under the various conditions. Yuan and Kendziorski (2006b) used a similar technique along with EM to cluster the genes while identifying them as equivalently or differentially expressed. This technique is similar to our approach of clustering genes according to how they vary between treatments. However, our approach finds (and clusters on) more complicated warps between genes under different treatments, rather than just how likely the genes are to be differentially expressed.

Ernst et al. (2005) clustered short time series by creating a representative set of distinct temporal expression profiles. These "model profiles" are chosen independently of the data. Each gene is then assigned to the profile to which it has the highest correlation. Significance tests are then done to see which profiles have a higher number of members than expected.

Leng and Müller (2006) used principal component analysis to perform clustering. The technique is to extract several orthogonal components from the time series, and express each gene as a linear combination of them. The genes are then separated into clusters based on the proportions of these components that are used. The method is related to that of Liu and Müller (2003), which is reviewed above. The method in this paper can also be used for clustering, based on the modes which it derives from the genes' expression levels.

Finally, Langmead et al. (2002b) were concerned with identifying genes with repeating expression patterns, which is a related task to clustering. In their formulation, the undirected Hausdorff

metric is used to calculate distances between genes. The authors claim that this better gauges their similarity than the correlation coefficient. The calculation is performed on the autocorrelation of each gene-expression time series, rather than on the series itself. This allows their similarity measure to be phase-independent. Langmead et al. (2002a) developed another variation of the algorithm, which uses maximum entropy to calculate a gene's phase and period.

3.3 Chapter Summary

In this chapter we have discussed the work of other groups that is related to our own. Those groups that explicitly align multiple time series have done the work very closely related to our own. Other groups that we have mentioned have modeled time-series data without explicitly aligning it, or have focused on particular tasks with which we are concerned as well.

Chapter 4

Interpolation of Gene-Expression Time Series

As has already been stated, gene-expression time-series data frequently suffer from non-uniform and irregular sampling. Very often, expression levels have been sampled only a few times over the course of the series. We thus use interpolation in order to reconstruct the missing data before computing alignments or doing other calculations. Linear interpolation would be an easy method to use, but B-splines have been used before successfully with similar gene-expression data (Bar-Joseph et al., 2003). We hypothesize that, in our case, B-splines can better reconstruct the data and filter out the noise inherent in microarray data. In pursuit of this, we have developed novel methods to fit observed data and reconstruct the intermediate times. We refer to these interpolated intermediate observations as *"pseudo-observations."*

B-splines are a kind of piecewise polynomial (Rogers and Adams, 1989). They generalize linear interpolation so that the reconstructed data can consist of quadratic, cubic, or higher-order curves. Further, B-splines are capable of acting as filters, eliminating extremes in the reconstructions which result from using noisy data as input.

This chapter will first define B-splines and derive their properties. We then discuss the techniques we use for them in representing gene-expression data. Finally, we show the experiments we have run to demonstrate their utility in this domain. Parts of the material covered in this chapter were originally published in Smith et al. (2008).

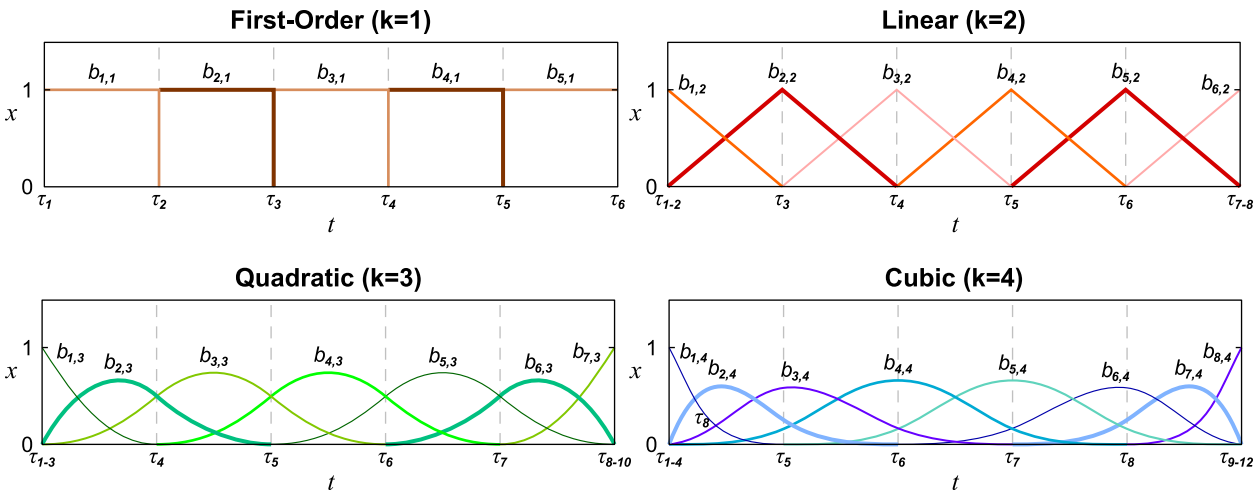

Figure 4.1: B-spline bases of various orders. The bases are defined by the given knots in conjunction with the Cox-de Boor equations (Equations 4.3 and 4.4). Knots (points of discontinuity) are at the dotted lines and the edges. Note the k-multiplicity of knots at the edges.

4.1 B-Splines

A B-spline is a piecewise polynomial function that is a generalization of a Bézier curve (Rogers and Adams, 1989). When working in two dimensions, there is an independent variable and a dependent variable. We refer to the independent dimension as t, and the dependent one as x. The spline is the weighted sum of a set of basis splines. These bases are a set of curves that are defined by two factors: the desired order k of the splines, and the vector $\vec{\tau}$ of points of discontinuity in the t dimension, which are called *knots*. Each knot is a number in t, and they are restricted so that each successive knot is greater or equal than its predecessor. The B-spline is defined only within the span of the knots:

$$\tau_1 \leq t \leq \tau_{|\vec{\tau}|}. \tag{4.1}$$

A B-spline contains n bases, where:

$$n = |\vec{\tau}| - k. \tag{4.2}$$

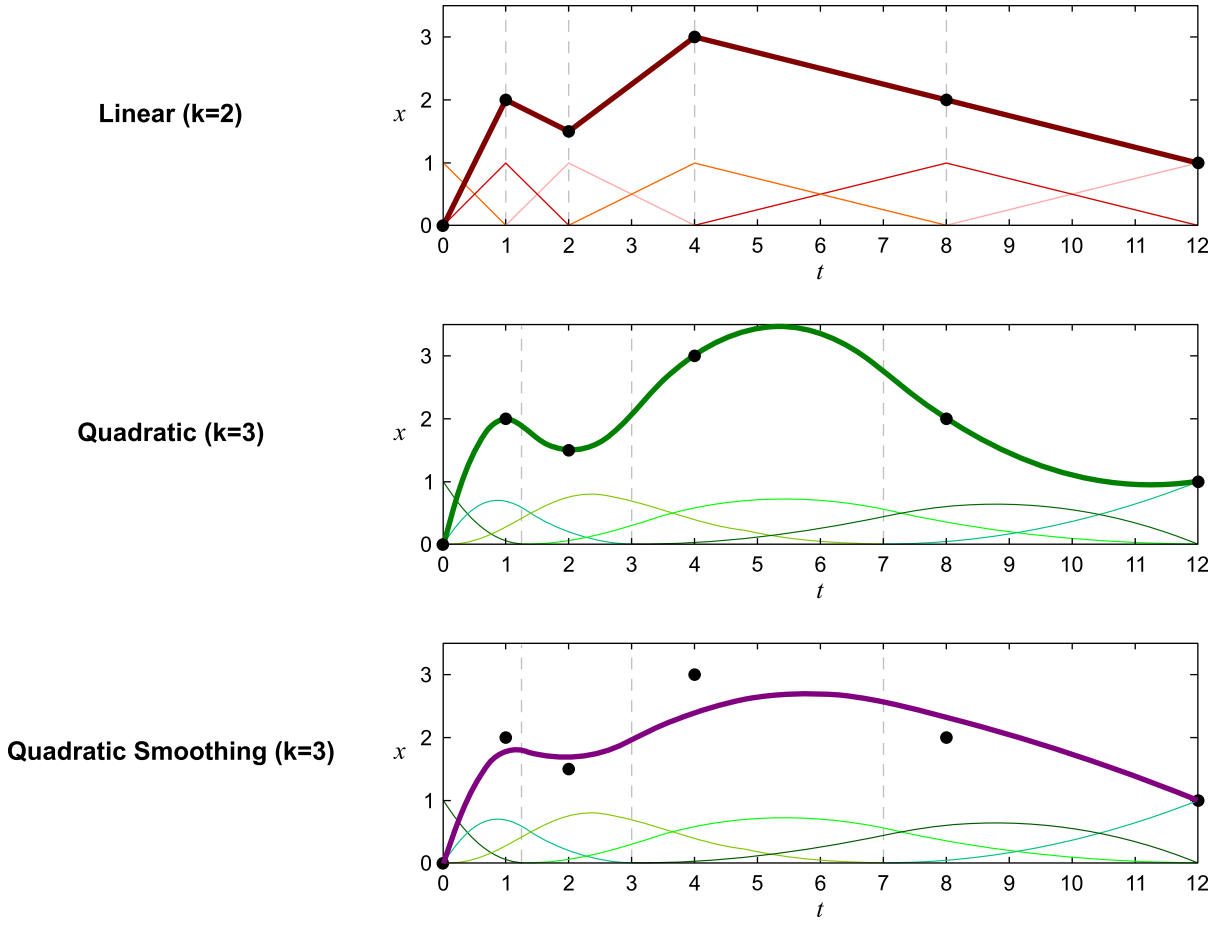

Figure 4.2: Figures of B-splines of various orders and types. The main spline which fits the observed points is a weighted sum of the basis splines, shown at the bottom of each panel. Knots, or points of discontinuity, are represented by vertical dotted lines. The linear and quadratic splines fit the points directly, while the smoothing spline uses the weights from a lower-ordered spline.

The basis splines are formally defined by the Cox-de Boor regression formulas:

$$b_{i,1}(t) = \begin{cases} 1 & \text{if } \tau_i \leq t \leq \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{4.3}$$

$$b_{i,k}(t) = \frac{t - \tau_i}{\tau_{i+k-1} - \tau_i} b_{i,k-1}(t) + \frac{\tau_{i\ +k} - t}{\tau_{i+k} - \tau_{i+1}} b_{i+1,k-1}(t), \tag{4.4}$$

where $b_{i,k}$ is the ith basis of order k. Figure 4.1 illustrates basis splines for orders one through four, while Figure 4.2 shows B-splines of orders two and three which have been calculated to fit observed points.

Several important properties emerge from these equations:

- Each basis has a degree of $k-1$. Thus a second-order basis consists of line segments, a third-order basis consists of quadratic segments, etc. Because the B-spline is a weighted sum of these bases, each segment of the spline inherits this property.
- A B-spline is continuous down to the $(k-2)$th derivative. Thus they appear to be a "natural" interpolation that is pleasing to the eye (Rogers and Adams, 1989).
- Each basis is only nonzero within the span of $k+1$ knots. This means that changing the weight of any one basis will alter the B-spline only in a limited area.
- The bases will sum to one at any given point.
- When the first knots do not coincide, points near the beginning of the spline's range are dependent on fewer bases than those in the middle. The same holds true at the end of the spline. Further, in such a case the spline must be zero at the ends, since all of the bases will have a value of zero there. These problems can be rectified by chosing knots so that k of them coincide at both ends of $\vec{\tau}$. This is called k-*multiplicity*, and is shown in Figure 4.1.

The interpolating B-spline s is formally defined as:

$$s(t) = \sum_{i=1}^{n} C_i b_{i,k}(t). \tag{4.5}$$

The weights C_i are known as control points. Since each basis is well defined over its length, solving for the control points is a simple matter of solving linear equations. If there are n points to interpolate, each consisting of a pair (t_i, x_i), we obtain:

$$\begin{bmatrix} b_{1,k}(t_1) & \cdots & b_{n,k}(t_1) \\ \vdots & \ddots & \\ b_{1,k}(t_n) & & b_{n,k}(t_n) \end{bmatrix} \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \tag{4.6}$$

or, more compactly:

$$\mathbf{b}\vec{C} = \vec{x}. \tag{4.7}$$

However, we must make sure that every basis overlaps in time with at least one observation, or the matrix will be rank-deficient and the equations unsolvable. With fewer than n points, the problem is underconstrained and cannot be solved as stated. In such a case, the order of the spline or the number of knots must be reduced. With more than n points, the problem is overconstrained and can only be solved in a least-squares sense. This is easy to do with standard linear algebra techniques, where we determine $\vec{C}$ by:

$$\mathbf{b}^T\mathbf{b}\vec{C} = \mathbf{b}^T\vec{x}. \tag{4.8}$$

Here, $\mathbf{b}^T$ is the transpose of $\mathbf{b}$.

4.2 Applying B-splines to Expression Data

When working with gene-expression time-series data, the independent variable t is time, and the single dependent variable x is expression. Obviously, the choice of knots will have a large effect on the interpolating B-spline calculated. Previous work (Bar-Joseph et al., 2003) has assumed uniform knots, and then fit the splines directly using Equation 4.7 or 4.8. However this can often lead to unsolvable equations when the data is too sparse, or to splines that oscillate too much while trying to fit the observed points exactly. We propose two methods in order to rectify these problems. First, we use the observed times themselves as the knots. Second, we use the control points from a lower order spline, which smooths out the function.

Consider the case in which there are observations at six, twelve, 24, and 48 hours. When there are five or more knots spread uniformly over this 48 hour range, a second-order basis spline will

have a span of 24 hours or less. It is very likely that one of the last bases will fall into the gap between the final two observations, making the matrix $\mathbf{b}$ rank-deficient. Our solution is to use the observed times themselves as the knots, with k-multiplicity at the ends (as explained above). This provides three notable advantages. First, it avoids the problem of rank-deficient matrices detailed above. Second, it means that the splines will be more concentrated in precisely those areas that the researchers who gathered the data chose to make their observations. Third, it means that the second-order linear spline reduces to simple linear interpolation between the observations. This is not only intuitive, but it makes more direct the comparison of our methods to this basic form of interpolation.

However, it is not possible to use precisely these knots when using a B-spline of third-order or greater, and still maintain k-multiplicity. Every time the order increases by one, the number of duplicate knots at the ends must increase by two. However n cannot increase, because it is bounded by the number of observations seen. Thus by Equation 4.2, as k increases by one, $|\vec{\tau}|$ may only increase by one as well, and not two. Thus, one of the unique knots must be eliminated.

We address this issue by resampling the knots so that they have roughly the same distribution. This can be seen in Figure 4.2, between the linear and quadratic splines. The number of knots has been reduced by one for the quadratic spline. Formally, the new knot vector $\vec{\tau}^*$ relates to the old one $\vec{\tau}$ as:

$$\tau_i^* = \tau_{\lfloor j \rfloor} + (j - \lfloor j \rfloor)(\tau_{\lfloor j \rfloor + 1} - \tau_{\lfloor j \rfloor}) \tag{4.9}$$

where j is the coordinate into $\vec{\tau}$ that corresponds to i in $\vec{\tau}^*$. j can be calculated from i by the following relation:

$$j = \begin{cases} k & \text{if } i \leq k^* \\ |\vec{\tau}| - k + 1 & \text{if } i > |\vec{\tau}^*| - k^* \\ \frac{|\vec{\tau}| - 2k + 1}{|\vec{\tau}^*| - 2k^* + 1}(i - k^*) + k & \text{otherwise} \end{cases}, \tag{4.10}$$

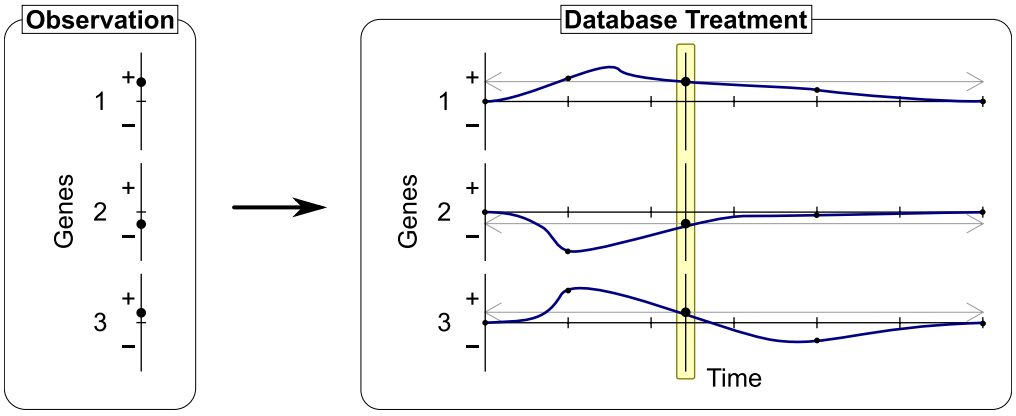

Figure 4.3: Comparison of an observation to an interpolated treatment. The observation is compared to all times in all treatments, to identify the time and treatment nearest to it. In this figure, the best answer is highlighted.

where k^* is the order of the new spline and k is the order of the old one. Note that the coordinate j will often not be an integer. In such a case, Equation 4.9 will linearly interpolate between τ_j and τ_{j+1} to find τ_i^*. For example, if j is 3.5, τ_i^* will be halfway between τ_3 and τ_4.

Another problem with B-splines is that they have a tendency to overfit curves in data-impovershed conditions. Such reconstructions can show large oscillations in an attempt to exactly intercept every observed data point. This can be especially problematic with microarray data, which are already inherently noisy. In order to rectify these problems, we solve for the control points of a low-order spline, and then use those for a higher-order one. This is similar to techniques illustrated in Rogers and Adams (1989). Such a spline will tend to fall within the convex hull created by the lower-order spline. We refer to such splines as *smoothing* splines, and refer to B-splines solved with conventional methods as *intercepting* splines. A smoothing spline is shown in the third panel of Figure 4.2. Note that the bases are identical to those in the quadratic panel, although the control points are those from the linear one.

4.3 Experiments

We wish to know which interpolation method (including simple linear interpolation) is the best to use to reconstruct missing gene-expression data. We do this by performing a leave-one-out experiment in which we classify each observation in our data set in turn, using the remaining observations as set against which to compare it.

The data we use in our experiment comes from the EDGE toxicology database (Hayes et al., 2005), and can be downloaded from `http://edge.oncology.wisc.edu/`. Our data set consists of 216 unique observations of microarray data, each of which represents the expression values for 1600 different genes. Each of these expression values is calculated by taking the average expression level from four treated animals, divided by the average level measured in four control animals. The data are then converted to a logarithmic scale, so that an expression of 0.0 corresponds to the average basal level observed in the control animals.

Each observation is associated with a *treatment* and a *time*. The treatment refers to the chemical to which the animals were exposed and its dosage. The time indicates the number of hours elapsed since exposure occurred. Times range from 6 hours up to 96 hours. The data used in our computational experiments span 11 different treatments, and for each treatment there are observations taken from at least three different time points. We can assume that for all treatments there exists an additional implicit observation at time zero. This is the time at which the treatment was applied, so all expression values are assumed to be at base level.

For each trial in our classification experiment, we refer to the left-out data point as the *query*, and the data being interpolated and searched for a match as the *database*. The task we consider is to identify the treatment and time in the database that provide the best match to the query. This is illustrated in Figure 4.3. Not only do we exclude the query observation from the database, but we also exclude all observations of the same time and treatment. This forces the program to use interpolation to find the correct answer; it must reconstruct pseudo-observations in order to find a match with the right time and treatment. We use every observation in turn as the query, except those whose time is the last one of their respective treatments. This is because our techniques

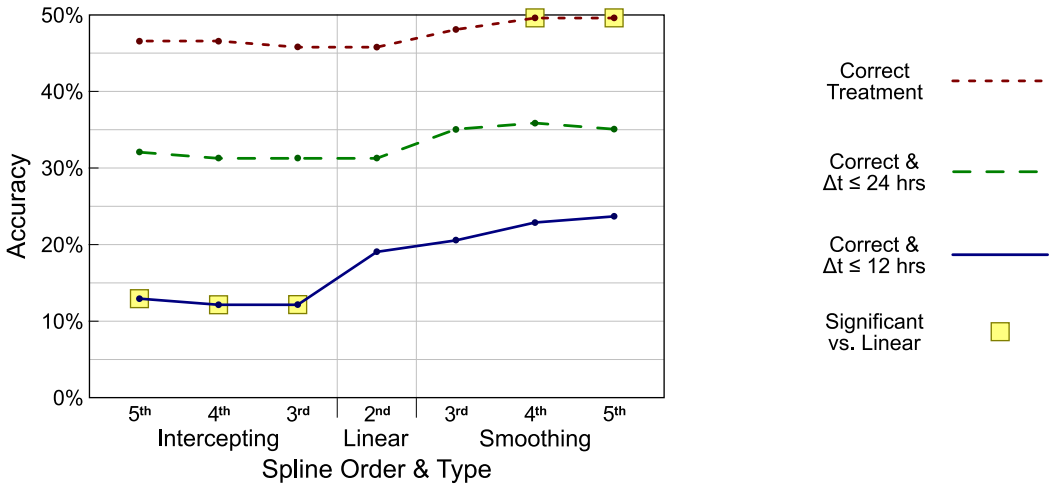

Figure 4.4: Treatment and alignment accuracies using different B-splines for interpolation. All replicates of the observation tested are purged from the database before interpolation. The top line shows treatment accuracy, in which the correct treatment is chosen. The bottom lines show alignment accuracy, where the predicted time is within 24 and 12 hours respectively of the actual time. Highlighted points are significantly different from the linear case ($p \leq 0.05$ via McNemar's χ^2 test).

cannot extrapolate matching times and treatments from the database for these observations. (We can interpolate matching times and treatments for the first observations of each treatment, because of the implicit observation at time zero.)

We reconstruct the pseudo-observations hourly for every treatment in the database, using the different methods of interpolation. We then classify the query. We wish to know how accurately we are able to (i) identify the treatment from which each point was extracted, and (ii) align each query point to its actual time in the time series for the treatment. We refer to the former as *treatment accuracy* and the latter as *alignment accuracy*.

We note that this task is only a surrogate for the actual task with which we are concerned—classifying uncharacterized chemicals and aligning them with the most similar treatment in a database. It is a useful surrogate, however, because it is a task in which we know the most similar treatment and the correct alignment of the query to this treatment.

The method we use to measure distance between the query observation and the treatment pseudo-observation being considered is a scale-independent Euclidean distance:

$$D_{SIE}(a,b) = \min_{u} \left(\sqrt{\sum_{g \in G} (a[g] - ub[g])^2} \right), \tag{4.11}$$

in which a and b are two observations, G is the set of genes (dimensions), and u is a scalar chosen by least-squares in order to minimize the distance. We use this algorithm in order to account for the differential scaling some observations may have. It is not uncommon for all the readings in one observation to be scaled up or down, due to technical limitations in the microarray process (Barenco et al., 2006).

We consider seven different interpolation methods in all. We look at both *intercepting* and *smoothing* splines as explained in Section 4.2, with orders three, four, and five. The control points for the smoothing splines are based on those for second-order interpolation. We also perform linear (i.e. second-order) interpolation as a control. If there are too few observation times for a particular order, we use the highest possible order. (For example, if there are two real observations and the vitual zero-observation, we cannot interpolate with an order higher than three.)

The results of this experiment are shown in Figure 4.4. The top line shows treatment accuracy, while the lower lines show alignment accuracy—where a case is considered "correct" if in addition to the proper treatment, the predicted time is correct to within 24 or 12 hours respectively. We test the significance of the differences in accuracy (from the linear interpolation control) using McNemar's χ^2 test. Highlighted points are those deemed significant, with $p < 0.05$. For all three accuracy measures we see improvement when using smoothing splines, while intercepting splines perform similarly or worse than the linear interpolation control. The fifth-order smoothing spline has a significantly higher classification accuracy ($p \approx 0.025$), and also appears to have better alignment accuracy ($p \approx 0.132$ for $\Delta t \leq 24$ and $p \approx 0.180$ for $\Delta t \leq 12$). By contrast the more traditional intercepting spline is likely overfitting its interpolation to the limited number of observed times. Although the fifth-order intercepting spline is not significantly different from the linear one for classification accuracy ($p \approx 0.739$) and alignment accuracy to within 24 hours ($p \approx 0.705$), there is a noticeable hit in the stricter alignment accuracy ($p \approx 0.021$). The p-values for the lower-ordered splines are qualitatively similar.

4.4 Summary

In this chapter we have reviewed the B-spline, which we use to interpolate missing data in order to reconstruct time series. We have also detailed certain techniques we use in order to adapt the B-spline to gene-expression time-series data:

- We use knot vectors with k-multiplicity in order to allow the B-spline to fit nonzero expression levels at its ends.
- We use the times of the observed points to define the knot vector. This keeps the linear equations associated with the B-spline solvable, and concentrates the points of discontinuity at the times the researchers thought interesting enough to measure.
- We developed a technique using smoothing splines in order to filter out noise from the sparse data and the expression measurement process. A smoothing spline is a spline whose control points are solved from a lower-order spline. We have detailed a method for resampling the knots to allow this.

Our experiments show that these methods have predictive value. Our reconstructed time series allow better classification and alignment of held-aside data than those obtained via linear interpolation or other B-splines.

Chapter 5

Alignment of Time Series Data

This chapter details novel methods to align two series in order to find correspondences of maximal similarity. Parts of the material covered in this chapter were originally published in Smith et al. (2008) and Smith et al. (2009).

5.1 Segment-Based Algorithms

Here we detail our two novel methods to align a pair of time series. As before, we refer to the series as $\vec{q}$ and $\vec{d}$, refering to a *query* series being aligned with a *database* series. However these names are conventions only.

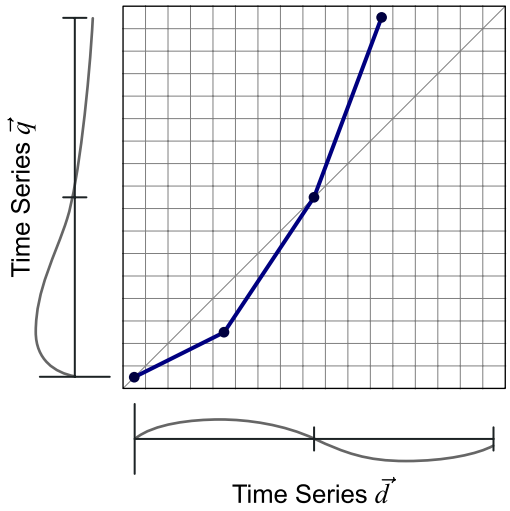

Figure 5.1: Segment-based time warping. The three segments each show correspondences between limited sections of the time series. The relation between $\vec{d}$ and $\vec{q}$ is different in each segment.

Like Correlation-Optimized Warping (COW), which we reviewed in Section 2.2.4, these alignment methods are *segment-based* methods. Figure 5.1 shows an example of a warping path obtained by such a method. The path consists of m segments, where possible values of m have been predefined. Likewise both $\vec{q}$ and $\vec{d}$ are partitioned into m sections, where the ith sections correspond to each other. (Due to shorting, one or both ends of one of the series may be excluded from this partitioning.) We call the points of discontinuity within the alignment path *knots*.

Segment-based methods take into account that the nature of the relationship between the two series may vary as they progress. Each segment is independent of the others. Thus the relationship between the ith segments may be completely different from the relationship between the $(i+1)$th segments. For example, it may be the case that the first part of the expression response occurs more slowly in one treatment than in a similar treatment. Often, the final score of the alignment is just the sum of the scores of each of the segments. (Sometimes there is an additional summand as well, such as in our multisegment generative algorithm.)

5.1.1 Multisegment Generative Alignment

Our *multisegment generative* method is the first approach we developed to align and compare a pair of time series. Each segment is scored based on how similar $\vec{q}$ and $\vec{d}$ are within its span. We use dynamic programming in conjunction with this model to find the best alignment path using up to M segments. The warp found is composed of discrete segments, like the warp shown in Figure 5.1.

We start by scoring a given alignment, in which we already know the locations of the segments and knots. To determine the similarity between a query time series $\vec{q}$ and a particular database series $\vec{d}$, we can calculate how likely it is that $\vec{q}$ is a somewhat distorted exemplar of the same process that resulted in $\vec{d}$. In particular, we can think of a generative process that uses $\vec{d}$ to generate similar expression profiles. We can then ask how probable $\vec{q}$ looks under this generative process.

Given this generative process idea, we calculate the probability of a particular alignment of query $\vec{q}$ given a database series $\vec{d}$ as follows:

$$P(\vec{q}, s, a|\vec{d}) = P_m(m) \prod_{i=1}^{m} P_s(s_i) P_a(a_i) P_e(\vec{q_i}|\vec{d_i}, s_i, a_i), \tag{5.1}$$

where m is the number of segments in the alignment, and $\vec{q_i}$ and $\vec{d_i}$ refer to the expression measurements for the ith query and database segments respectively. P_m represents a probability distribution over the number of segments in an alignment, up to some maximum number M of allowed segments. P_s represents a probability distribution over possible stretching values for a pair of segments, P_a represents a probability distribution over possible amplitude values, and P_e represents a probability distribution over expression observations in the query series, given the database series and the stretching and amplitude parameters.

The values s_i and a_i are called the *stretching* coefficient and the *amplitude* coefficient. They can be calculated directly from $\vec{q_i}$ and $\vec{d_i}$. Stretching refers to distortions in the rate of response. The stretching coefficient is just the ratio between the lengths of the series segments:

$$s_i = \frac{|\vec{d_i}| - 1}{|\vec{q_i}| - 1}, \tag{5.2}$$

where $|\vec{x}|$ is the number of elements in $\vec{x}$ (making $|\vec{x}| - 1$ the difference in index between the last and first elements). Amplitude refers to how much the expression values vary from the mean. It is defined as:

$$a_i = \sqrt{\frac{\sum_{j=0}^{|\vec{d_i}|-1} \vec{d_i}(j)^2}{\sum_{j=0}^{|\vec{q_i^*}|-1} \vec{q_i^*}(j)^2}}, \tag{5.3}$$

where $\vec{q_i^*}$ is the series segment $\vec{q_i}$ that has been resampled uniformly to have the same length as $\vec{d_i}$, and $\vec{d_i}(j)$ and $\vec{q_i^*}(j)$ refer to expression values at specific times within $\vec{d_i}$ and $\vec{q_i^*}$.

For the experiments we report here, we use a uniform distribution for P_m over all legal values of m:

$$P_m(m) = \begin{cases} 1/M & \textbf{if } 1 \leq m \leq M \\ 0 & \textbf{otherwise} \end{cases}. \tag{5.4}$$

To represent P_s, we use a discretized version of the following distribution:

$$P(x) = \frac{e^{\frac{-\sigma^2}{2}}}{\sigma\sqrt{2\pi}} \times e^{\frac{-log^2 x}{2\sigma^2}}. \tag{5.5}$$

We choose this distributional form because it is a variation of the log normal distribution that is symmetric around one, such that $P(x) = P(1/x)$. Thus for example, stretching some expression response by a factor of two is equiprobable to compressing it by a factor of two. This symmetry property means that it does not matter which series we consider to be the query and which we consider to be from the database. As we discuss below, our dynamic programming algorithm only allows segments to begin and end at a limited number of points. Thus, our distribution is actually discretized so that probability mass is allocated only to possible stretching values, and then renormalized.

We use a similar distribution to represent P_a, the distribution of amplitude values, since we also want to have $P(x) = P(1/x)$ symmetry with these values. Thus a twofold increase in an expression response is treated as equiprobable to a twofold decrease.

To calculate $P_e(\vec{q_i}|\vec{d_i}, s_i, a_i)$, we transform our representation of $\vec{d_i}$ using the given stretching and amplitude values, and then ask how probable $\vec{q_i}$ appears when we use this transformed $\vec{d_i}$ series as a model. Let us first consider a simple case in which our time series have only one gene, and we are mapping only one point from the query segment $\vec{q_i}$ to the database segment $\vec{d_i}$. Let t represent a time coordinate in the segment $\vec{q_i}$, and let k_i^q and k_i^d represent the ith knots with respect to $\vec{q}$ and $\vec{d}$, respectively. (Thus, k_i^q represents the earliest time in $\vec{q}$ within segment i, and likewise for k_i^d and $\vec{d}$.) Then we can map a time coordinate from segment $\vec{q_i}$ into the corresponding coordinate in $\vec{d_i}$ as follows:

$$t' = k_i^d + (t - k_i^q) \times s_i \tag{5.6}$$

where the stretching value is defined as above in Equation 5.2.

Our model for "generating" points in the query series from a point in the database series is a Gaussian centered at the database point. Let $p(x;\ \mu, \sigma_e)$ represent the probability density function of this Gaussian, where μ is the mean and σ_e is the standard deviation of the Gaussian. We can then compute the probability of generating a query point $\vec{q_i}(t)$ located at time t as:

$$p\big(\vec{q_i}(t);\ a_i \times \vec{d_i}(t'), \sigma_e\big). \tag{5.7}$$

In other words, we center a Gaussian on the expression level at the mapped time coordinate in the database series, and ask how probable the scaled expression value from the query looks at that time coordinate.

To generalize this calculation to multiple observations in the query series, we make the simplifying assumption that the observations are independent, and we have:

$$P_e(\vec{q_i}|\vec{d_i}, s_i, a_i) \propto \prod_{j=1}^{n_i} p\big(\vec{q_i}(t_j);\ a_i \times \vec{d_i}(t'_j), \sigma_e\big) \tag{5.8}$$

where n_i is the number of query observations in segment i.

Each of our observations represents measurements for hundreds of genes. We therefore generalize the description above by having $p(x;\ \mu, \sigma_e)$ be a multidimensional Gaussian, with one dimension for each gene measured. We treat the genes as independent of one another given the time point. Thus the covariance matrix for this Gaussian is zero on all of the off-diagonal terms.

We assume that σ_e represents variation in expression measurements that are due to technical and biological variability. Thus, we estimate the standard deviation for each gene by considering the variance in a sample that consists of all the replicated experiments in the database.

In addition to considering the likelihood of the query series under the assumption that it exhibits a similar response to the given database series, we also consider its likelihood under a null model. The notion of a null model here is one that generates alignments by randomly picking observations from the database to align with the query sequence. The rationale for using such a null model is analogous to the use of a model of *unrelated* sequences in the derivation of substitution matrices for protein sequence alignment (Altschul, 1991; Durbin et al., 1998). In the case of protein sequence alignment, we want to know the relative likelihood of two cases: one case in which the correspondence between the sequences is explained by their relatedness through evolution, and the alternative in which the sequences are unrelated. In our task, we similarly want to compare the probability of an alignment given a model of relatedness (described above), and an alternative that asks how probable the query would look if we aligned it to an unrelated series.

The value of a null model for our application is that it enables alignments of differing lengths, including shorted alignments, to be compared on an equal footing. Under our scoring function

which incorporates the null model, segments have a positive score only if the database series in that segment explains the corresponding segment from the query series better than the null model does.

Let $p(x; \mu_{DB}, \sigma_e)$ represent the probability density function of a multidimensional Gaussian whose mean μ_{DB} is the average expression level of the observations in the database, and whose standard deviation is σ_e as before. We then estimate the probability of the ith segment of the query series under the null model as:

$$P_{\text{null}}(\vec{q_i}) \propto \prod_{j=1}^{n_i} p\big(\vec{q_i}(t_j); \mu_{DB}, \sigma_e\big). \tag{5.9}$$

Since our null model assumes that there is only a single segment with no amplitude change or stretching, we can compute the probability of the entire query series $\vec{q}$ as follows:

$$P_{\text{null}}(\vec{q}) \propto \prod_{i=1}^{m} \prod_{j=1}^{n_i} p\big(\vec{q_i}(t_j); \mu_{DB}, \sigma_e\big). \tag{5.10}$$

Putting together the terms above, we can score a given alignment based on the log of the likelihood ratio of the query series under the "database series" model versus the query series under the null model as:

$$\text{score}(\vec{q}, \vec{d}) = \log P_m(m) + \sum_{i=1}^{m} \left(\log P_s(s_i) + \log P_a(a_i) + \log P_e(\vec{q_i}|\vec{d_i}, s_i, a_i) - \log P_{\text{null}}(\vec{q_i}) \right) \tag{5.11}$$

Up to now we have described this process in terms of using a database series to generate the query series. However, we want our alignment method to be symmetric so that it does not matter which series we consider to be the query and which we consider to be from the database. Due to the last two terms, this will not necessarily be the case using the scoring function defined above. Therefore, we modify the scoring function so that it also considers using the query series to generate the database series:

$$\text{score}(\vec{q}, \vec{d}) = \begin{array}{c} \log P_m(m) + \sum_{i=1}^{m} \Big(\log P_s(s_i) + \log P_a(a_i) + \log P_e(\vec{q_i}|\vec{d_i}, s_i, a_i) \\ - \log P_{\text{null}}(\vec{q_i}) + \log P_e(\vec{d_i}|\vec{q_i}, {}^{1}\!/_{s_i}, {}^{1}\!/_{a_i}) - \log P_{\text{null}}(\vec{d_i}) \Big) \end{array} . \tag{5.12}$$

Here $P_e(\vec{d}_i|\vec{q}_i, {}^1\!/_{s_i}, {}^1\!/_{a_i})$ is calculated in an analogous manner to $P_e(\vec{q}_i|\vec{d}_i, s_i, a_i)$ but the inverses of s_i and a_i are used to generate observations in the database series.

Now that we have developed the scoring algorithm for a given alignment, we want to find the best alignment possible between a pair of series $\vec{q}$ and $\vec{d}$ with discrete observations. We do this using dynamic programming. We start by defining the score for a single segment, which can be defined by the pair $(\vec{q}_i, \vec{d}_i)$. The following is a single summand from Equation 5.12:

$$\text{score}(\vec{q}_i, \vec{d}_i) = \begin{array}{l} \log P_s(s_i) + \log P_a(a_i) + \log P_e(\vec{q}_i|\vec{d}_i, s_i, a_i) \\ \quad + \log P_e(\vec{d}_i|\vec{q}_i, {}^1\!/_{s_i}, {}^1\!/_{a_i}) - \log P_{\text{null}}(\vec{q}_i) - \log P_{\text{null}}(\vec{d}_i) \end{array} . \tag{5.13}$$

The arguments to this scoring function are defined the same as above.

The core of the dynamic program involves filling in a three-dimensional matrix Γ in which each element $\gamma(i, x, y)$ represents the best score found with i segments that align the query subseries from time 0 to x with the database subseries from time 0 to y. The values x and y must be selected from the given observations in the two series. The basic idea is that in order to determine $\gamma(i, x, y)$, we look through all $\gamma(i-1, a, b)$ where $a < x$ and $b < y$. We then add the score of the segment from (a, b) to (x, y) to the value $\gamma(i-1, a, b)$, assigning the best such sum to $\gamma(i, x, y)$.

We define $\gamma(i, x, y)$ with the following recurrence relation:

$$\gamma(i, x, y) = \max_{a<x,\ b<y} \begin{cases} \log P_m(i) + \gamma(i-1, a, b) & \\ \qquad + \text{score}(a, x, b, y) & \textbf{if } x = |\vec{q}| - 1 \textbf{ or } y = |\vec{d}| - 1 \\ \gamma(i-1, a, b) + \text{ score}(a, x, b, y) & \textbf{otherwise} \end{cases} . \tag{5.14}$$

where the base case is:

$$\gamma(1, x, y) = \begin{cases} \log P_m(1) + \text{score}(0, x, 0, y) & \textbf{if } x = |\vec{q}| - 1 \textbf{ or } y = |\vec{d}| - 1 \\ \text{score}(0, x, 0, y) & \textbf{otherwise} \end{cases} . \tag{5.15}$$

Here, $|\vec{q}|$ and $|\vec{d}|$ are the lengths of the time series, and one less than one of these values is the last time of the series. The first condition in each recurrence relation ensures that the distribution over the number of segments P_m is taken into account when we consider the last pair of segments in a candidate alignment.

Recall that we are interested in possibly shorting the alignment, thus finding a local alignment rather than a global one. Allowed alignments are those that explain the entire extent of at least one of the two given time series. In order to recover the optimal alignment, we use a traceback procedure that involves scanning the elements of Γ that represent alignments that include the entirety of the query series, the entirety of the database series, or both, in a manner similar to our variation of dynamic time warping (detailed in Section 2.2.3). We find the end point of the best alignment with the following equation, and start the traceback from there:

$$\text{bestscore} = \max_{i,\ a \leq |\vec{q}|-1,\ b \leq |\vec{d}|-1} \Big(\gamma(i, a, |\vec{d}| - 1),\ \gamma(i, |\vec{q}| - 1, b) \Big). \tag{5.16}$$

Note that the algorithm as described here shorts only at the ends, and not at the beginnings of the input time series. Although it is possible to double-short using the multisegment generative algorithm, we have not done so for the experiments we report.

This dynamic program can be thought of as having three key "penalty terms" that determine the relative scores of alignments. These penalty terms correspond to the probability distributions that govern (i) the number of segments, (ii) the stretching values, and (iii) the amplitude values used in an alignment.

Preferences for the number of segments to be used in alignments are expressed by providing a distribution for P_m. In our work to date, we have assumed a uniform distribution up to the allowed number of segment pairs. It might be valuable to use a distribution that favors fewer segment pairs, however. Preferences for stretching and amplitude values are controlled via the standard deviation σ parameter in the distributions over these values. For example, as σ_a for the amplitude distribution is made smaller, a difference in amplitude between the series is penalized more in the scoring scheme.

The method is more accurate than previous methods with respect to finding similar series and aligning them, but it is computationally intensive. The time complexity of this algorithm is $O(n^5)$, where n is the length of one of the time series. Calculating the score for a single segment has a complexity of $O(n)$, and all possible segments must be calculated. Each one is defined by a starting and ending location, in two dimensions. This gives us $O(n^4)$ segments, and so the total time complexity is $O(n^5)$. (We must also fill the matrix Γ, whose size is $M \times n \times n$ where M is the

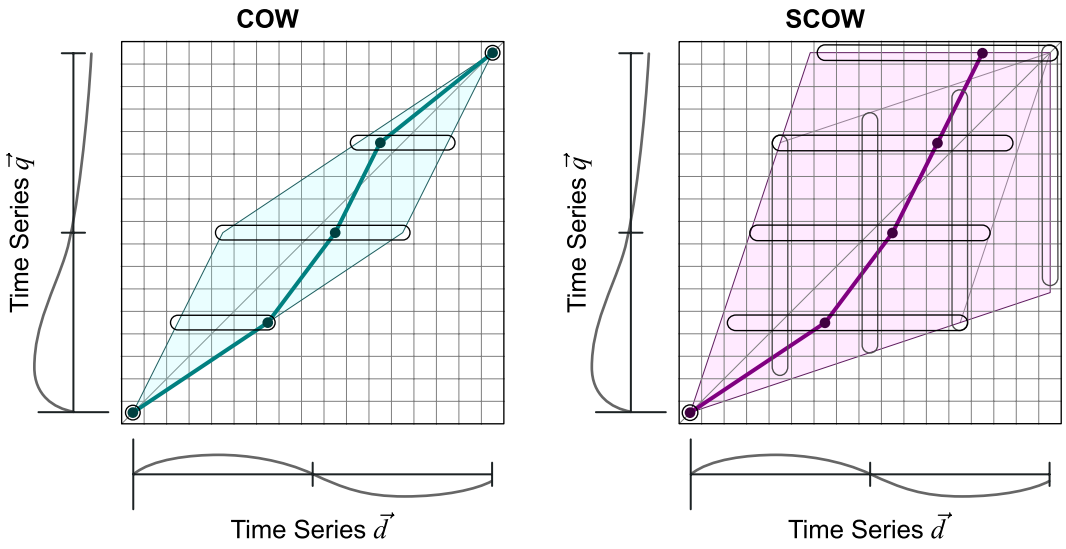

Figure 5.2: Correlation-optimized warping vs. shorting correlation-optimized warping. COW assumes a global alignment, and only searches for good knots with respect to one series ($\vec{d}$). By contrast, SCOW can short, and searches for good knots with respect to both series.

maximum number of segments. However by caching segment scores, filling Γ is decoupled from the segment score calculation. This makes the time complexity $O(n^5 + Mn^2)$, but n^5 dominates this calculation.) Each series-to-series comparison can take as much as several minutes, meaning that comparing a single query against a whole library of series may take prohibitively long.

5.1.2 Shorting Correlation-Optimized Warping

SCOW, or shorting correlation-optimized warping, is another segment-based method that we developed to align a pair of time series. It is a successor to the COW (correlation-optimized warping) algorithm (Nielsen et al., 1998), on which it is based. The two algorithms are contrasted in Figure 5.2. As explained in Section 2.2.4, COW has several limitations when working with gene-expression data:

- COW cannot short—it forces a global alignment of the time series.

Table 5.1: Pseudocode for SCOW. Knots are recalculated at least three times. The Traceback function extracts the best knots found from the previous pass to use in the next one.

procedure *SCOWAlign(series* $\vec{d}$*, series* $\vec{q}$*, set of genes* G*)*:

// initial passes //

$K^q \leftarrow$ *evenly spaced integers from* 0 *to* $|\vec{q}|$
$K^d \leftarrow$ *evenly spaced integers from* 0 *to* $|\vec{d}|$
calculate Γ^q,Γ^d *using* G
if (*BestScore*$(\Gamma^q) >$*BestScore*(Γ^d))**:** $\vec{\alpha} \leftarrow \vec{q}$
else: $\vec{\alpha} \leftarrow \vec{d}$
$K^\alpha \leftarrow$*Traceback*(Γ^α)

// main loop //

repeat:
  swap-dimension $\vec{\alpha}$
  calculate Γ^α *using* G
  calculate BestScore(Γ^α)
  $K^\alpha \leftarrow$*Traceback*(Γ^α)
until K^q,K^d *converge*

- COW only searches for knots with respect to one of the input series, making the assumption that the knots are evenly distributed with respect to the other one.
- COW is apt to align segments which differ greatly in magnitude because it scores using the Pearson cross correlation.
- Using the Pearson correlation also means that COW may sometimes return an undefined value if the input series sections do not have a defined correlation (as when one or both consist of all zeros).

SCOW overcomes these limitations by performing multiple searches for the best knots using a different search procudure than COW, and by incorporating elements from the multisegment generative warping algorithm into its scoring function.

Figure 5.3 illustrates the steps taken by SCOW. It first performs two independent searches: one searching for knots with respect to $\vec{q}$ while assuming they are equally distributed in $\vec{d}$, and one

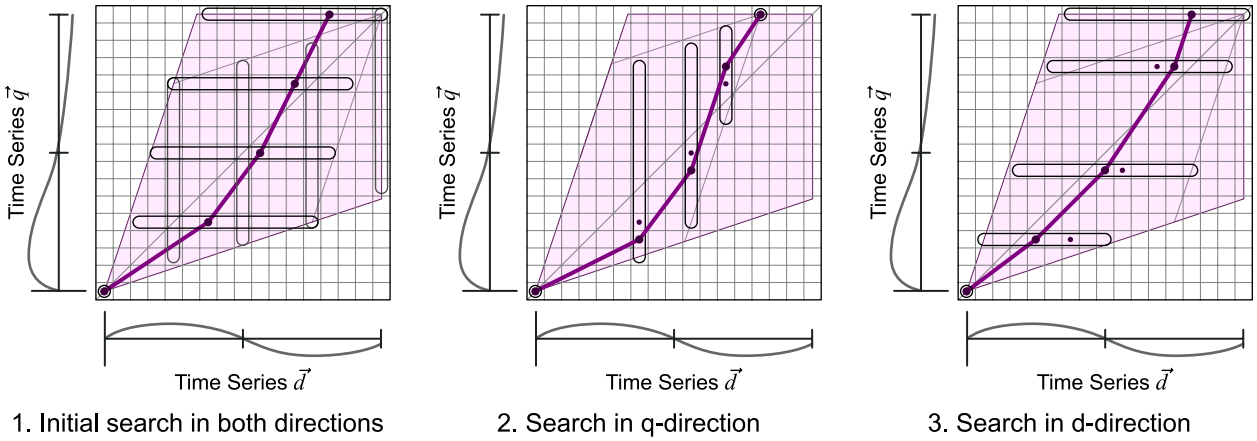

Figure 5.3: Steps taken by SCOW. The first step is to search for the best knots with respect to $\vec{q}$ while holding them constant in $\vec{d}$, and vice versa, using the best results found. Then we alternate searching in $\vec{q}$ and in $\vec{d}$ (using the knot locations found in the last step as the starting point) until convergence.

searching in $\vec{d}$ while assuming they are equally distributed in $\vec{q}$. Note that the algorithm can easily short in the dimension being searched, but not in the one being held constant. Any given set of knots determines an alignment, which can be scored as discussed below. After these initial two searches, the algorithm takes the best set of knots found overall, and begins to alternate searching with respect to each time series. It uses the best knots from the last search as the knots for the series currently being held constant.

For example, let us say that in the first two searches it found better knots when searching with respect to $\vec{d}$, as in the first panel of the figure. The next step would be to search in $\vec{q}$, using the knot locations found in the last step to anchor the knots in $\vec{d}$. It can then search in $\vec{d}$, using the last step to anchor them in $\vec{q}$ again. This alternation continues until the search converges. Because last best answer is necessarily part of each search, the set of knots found in each step must be at least as good as those found in the last step. Thus, the search is guaranteed to converge. Pseudocode for this algorithm is given in Table 5.1.

Whereas COW calculates elements of a single matrix Γ, SCOW uses two: Γ^q, which allows knots to vary with respect to $\vec{q}$, and Γ^d, which allows them to vary with respect to $\vec{d}$. The matrix Γ^α is of size $m + 1 \times |\vec{\alpha}|$, where m is the number of segments and α represents either q or d. Each

element $\gamma^{\alpha}(k,i)$ represents the best path found using k segments up to time i of $\vec{\alpha}$. Both matrices are initialized in the same way as Γ in COW, specified in Equation 2.9:

$$\gamma^{\alpha}(0,j) = \begin{cases} 0 & \textbf{if } j = 0 \\ -\infty & \textbf{otherwise} \end{cases}. \tag{5.17}$$

The remaining elements of the two matrices are calculated with the following two equations, which are analogous to Equation 2.10:

$$\gamma^{q}(k,i) = \max_{j \in \mathrm{pred}(i,k)} \left[\gamma^{q}(k-1,j) + \mathrm{score}\big(d(K^{d}_{k-1}, K^{d}_{k}), q(j,i)\big) \right], \tag{5.18}$$

$$\gamma^{d}(k,i) = \max_{j \in \mathrm{pred}(i,k)} \left[\gamma^{d}(k-1,j) + \mathrm{score}\big(d(j,i), q(K^{q}_{k-1}, K^{q}_{k})\big) \right]. \tag{5.19}$$

The matrix Γ^{q} is calculated only when one searches for knots with respect to $\vec{q}$ and holds them constant with respect to $\vec{d}$, while Γ^{d} is only calculated during the opposite case. The vectors K^{q} and K^{d} represent the coordinates of the knots in each dimension. The predecessor function $\mathrm{pred}(i,k)$ defines valid starting locations for a segment that ends at i, with respect to the dimension that is allowed to vary:

$$\mathrm{pred}(i,k) = \max\left[i - \lambda(K_k - K_{k-1}), \frac{1}{\lambda} K_{k-1}) \right], ..., \min\left[i - \frac{1}{\lambda}(K_k - K_{k-1}), \lambda K_{k-1}) \right], \tag{5.20}$$

where λ is the maximum slope allowed the aligning path in alignment space. This predecessor function is analogous to the "slack factor" used by COW.

In order to find the best score for a given pass, SCOW searches the last row of the matrix calculated for the maximum score, assuming the other dimension is not already shorted:

$$\mathrm{bestscore}(\Gamma^{\alpha}) = \begin{cases} \max\limits_{j} \gamma^{\alpha}(m,j) & \textbf{if } \text{other dimension not shorted} \\ \gamma^{\alpha}(m, |\vec{\alpha}| - 1) & \textbf{otherwise} \end{cases}. \tag{5.21}$$

This equation replaces Equation 2.13, which forces a global alignment. Importantly, Equation 5.21 prevents SCOW from shorting in both dimensions simultaneously. If the dimension not being searched is already shorted, SCOW does not search the last row of the Γ^{α} for better scores. Such a

case can be seen in the center panel of Figure 5.3. Because $\vec{d}$ is already shorted, we do not want to short $\vec{q}$ as well. The path is restricted so that it must end at the top of the matrix, aligning all of $\vec{q}$ to a portion of $\vec{d}$.

The scoring algorithm we define for SCOW takes aspects from the multisegment generative algorithm described in the previous section. In particular, the scoring function includes terms that incur penalties for segments that involve *stretching* and significant differences in *amplitude*, as were defined above.

We define the score of an alignment segment to be:

$$\text{score}(\vec{q_i}, \vec{d_i}) = \text{cor}(\vec{q_i}, \vec{d_i}) - \frac{\log^2 s_i}{2\sigma_s^2} - \frac{\log^2 a_i}{2\sigma_a^2}. \tag{5.22}$$

Here q_i and d_i denote the ith subseries of series $\vec{q}$ and $\vec{d}$ respectively, s_i is the amount of stretching in the alignment of the ith segments, a_i is the amplitude difference, and $\text{cor}(a, b)$ is the Pearson cross correlation. The values s_i and a_i are functions of $\vec{q_i}$ and $\vec{d_i}$. Stretching refers to distortions in the rate of response, and is just the ratio of the lengths of the two subseries. Amplitude is the difference in how much the gene expressions vary from their means, calculated via a least squares method. They are explicitly defined above in Equations 5.2 and 5.3. The terms containing s_i and a_i in Equation 5.22 are each just the logarithm of the probability distribution in Equation 5.5 (without the normalizing constants, which would have no effect on the calculation). The Pearson cross correlation (Coolidge, 2006) is the well known formula:

$$\text{cor}(\vec{a}, \vec{b}) = \frac{\sum_{i=0}^{|\vec{a}|-1}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{(\sum_{i=0}^{|\vec{a}|-1}(a_i - \bar{a})^2)(\sum_{i=0}^{|\vec{b}|-1}(b_i - \bar{b})^2)}}. \tag{5.23}$$

We also alternately add and subtract a tiny value ϵ to values in $\vec{q_i}$ and $\vec{d_i}$, so that correlation is always defined and two subseries with constant values will have a correlation of one.

SCOW operates with a time complexity of $O(imn^3)$, where i is the number of iterations performed, m is the number of segments, and n is the length of one of the interpolated series to be aligned. Each segment score can be calculated in $O(n)$ time. The number of predecessors for any one element of one of the matrices is also $O(n)$. The size of a matrix is $O(mn)$. With i

iterations, that gives the total of $O(imn^3)$. Since i and m are usually small numbers, this yields much better performance than the $O(n^5)$ of the multisegment generative method. Further, many of the calculations in successive passes of SCOW are the same, and may be cached. The speedup is dramatic: what takes the multisegment generative method an hour to calculate takes SCOW only a few seconds.

5.2 Experiments

We now wish to evaluate our two algorithms in order to asses their accuracy in aligning time series, and their ability to identify the most similar database series to an "unknown" query series. This section explains the methods we use in order to do this.

5.2.1 Data

As in Chapter 4, we use the EDGE toxicology database (Hayes et al., 2005) to evaluate our methods. This data set contains 216 unique observations of microarray data, each of which represents the expression values for 1600 different genes observed at a given *time* after a given *treatment* has been applied. Each of these expression values is calculated by taking the average expression level from four treated animals, divided by the average level measured in four control animals. The data are then converted to a logarithmic scale, so that an expression of 0.0 corresponds to the average basal level observed in the control animals.

The treatment refers to the chemical to which the animals were exposed and its dosage. The time indicates the number of hours elapsed since exposure occurred. Times in this data set range from six hours up to 96 hours. The data used in our computational experiments span eleven different treatments, and for each treatment there are observations taken from at least three different time points. We can assume that for all treatments there exists an additional implicit observation at time zero. This is the time at which the treatment was applied, so all expression values are assumed to be at base level.

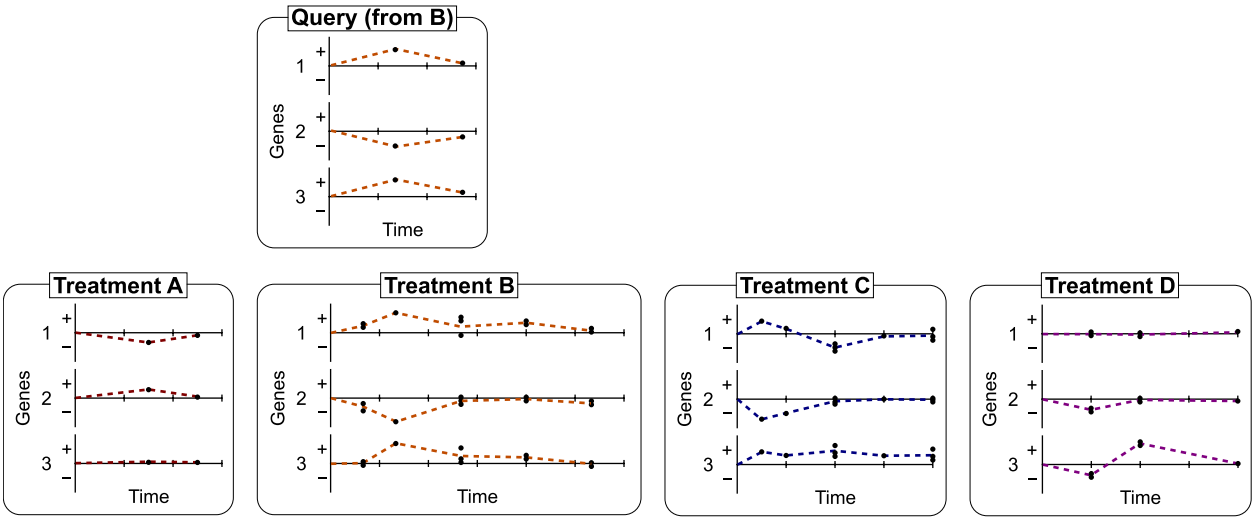

Figure 5.4: Separation of data into query and database. Treatment B is chosen, and a small number of observations are removed from it to form the query. The query can then be aligned and scored with all the database treatments.

5.2.2 Experimental Methodology

We wish to know how accurately our methods (i) identify the series in the database that is most similar to the query, and (ii) align each point from the query series to the appropriate point in the found series. As in the experiments in the last chapter, we call the former *treatment accuracy*, and the latter *alignment accuracy*.

Note that the first task is not a classification problem, but a similarity search. We are not looking for an exact match in the database, but a near match. In the real world, we will seldom perform a similarity search on a treatment that is already well-characterized. The purpose behind the similarity search is to help elucidate the properties of uncharacterized treatments. If we already understand the properties of the treatment being queried, the similarity search will not tell us anything new.

Ideally, we would test our algorithms on a data set in which each treatment has a well-defined nearest treatment. For example, we could query Treatment A1, and hope that the algorithm passes over Treatments B1, B2, C1, C2, etc., and returns Treatment A2. Unfortunately we do not have

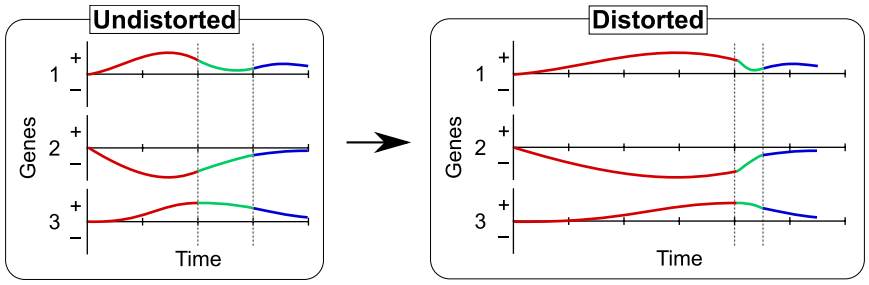

Figure 5.5: Adding distortion to a query. Here, the times for the first segment are doubled, those of the second segment are halved, and subsequent times remain undistorted (although they are translated in time to make room for the previous distortions).

such a data set. Instead we use the EDGE data set, and perform some special operations on it to make it a more useful surrogate.

We create our queries by separating them out from the database, as shown in Figure 5.4. Here, a few observations have been removed from Treatment B to act as the query. We then perform our similarity search, using all the treatments as the database. In addition to this, we sometimes *distort* the query temporally, by stretching or contracting segments of the treatment. This is illustrated in Figure 5.5. This creates a query that does not precisely match any treatment seen in the database, but which still has a ground-truth "correct answer" we can use to assess accuracy.

More specifically, we build each query set of a given treatment by first selecting the number of observations to be in it, then choosing which time points are represented, and finally picking an observation for each of these time points. The query sizes are chosen from a uniform distribution that ranges from one up to the number of observed times in the given treatment. The maximum size of a query set in the EDGE data is eight, although most consist of four or fewer observations. The time points are chosen uniformly as are the observations for each chosen time. Thus for each trial we have one query set, and eleven database sets formed from the remaining data. We perform 110 such trials: ten from each of our eleven treatments.

We then use smoothing splines (see Section 4.2) on each of these sets in order to reconstruct complete times series. We sample the time series at every four hours, so that we have regularly sampled *pseudo-observations* for all of the treatments in the database, as well as for the query. We

vary the order of the interpolating splines from two (linear) to five (quartic). In cases where the set has fewer times than the warranted order of spline, we use the maximum order spline possible. So if there are only three observations (including the virtual observation at time zero) and the trial calls for a fourth-order or fifth-order spline, we use a third-order one instead.

We use three different distortions, in addition to the undistorted case. The first one doubles all times in the first 48 hours (i.e., it stretches the first part of the series), and then halves all times (plus an offset for the doubling) for the next 24 hours. The second distortion halves for the first 36 hours and then doubles for 60 hours. The third one triples for the first 60 hours and then thirds for another 20. It should be noted that not all the treatment observations extend this long in time. The short ones (e.g. those for which we only have measurements up to 24 or 48 hours) will thus not be distorted as much as the long ones.

We can then test the various algorithms for treatment and alignment accuracy. We align the query in turn to each of the eleven database series, designating the series with the best score as the closest series. We then calculate the percentage of queries assigned the closest series from the same treatment. In addition, we record the average alignment error for each of the times for which there is a real query observation (not a pseudo-observation). For example, if the real query observations were taken at 24 hours, 36 hours, and 72 hours, and those three times were aligned respectively to twelve hours, 30 hours, and 72 hours, the average alignment error would be six hours.

We graph the results of these studies in figures such as Figure 5.6. The top line for each algorithm shows the treatment accuracy. The center line adds the criterion that in order to be a correct answer, the average alignment error must be less than or equal to 24 hours. For the bottom line, this is decreased to twelve hours. Points that are highlighted with squares are significantly different from some other method (which is specified in the particular figure). To determine significance, we use McNemar's χ^2 method, with a $p \leq 0.05$. We graph experiments in which we distort the query separately from those in which we do not.

One concern is that by adding distortion we could be changing the database treatment nearest to a given treatment. For example, maybe we would distort 10 $^{\mu g}/_{kg}$ of TCDD in exactly the right way to make it look like 64 $^{\mu g}/_{kg}$. To address this concern, we have performed similar distortion

experiments in which we align a distorted query series only to the database series that was used to generate it. The results of these experiment are qualitatively similar to the results reported here.

5.2.3 Comparison of Alignment Methods

We start by comparing several different alignment methods, estimating both treatment and alignment accuracy. The results of these experiments are shown in Figure 5.6. We use the methodology explained in the last section.

We test the multisegment generative method, SCOW, as well as several control methods. We test the multisegment method twice: first with M being set to three, and then with it set to one. For the three-segment generative method, we set the parameters of this method as follows. We set the probability that the model has one, two, or three segments at $1/3$ each, and 0 beyond that. We estimate σ_e (the deviation of the expression Gaussian) to be the standard deviation of the known observations as described previously. We set both σ_s (the stretching deviation) and σ_a (the amplitude deviation) to be $(\#\ genes)^{-1}$. Thus the three main components of the model have roughly equal influence. For the one-segment generative method, we set the parameters the same except that the probability of there being one segment is 1, and 0 beyond that. For SCOW, we set both σ_a and σ_s to be 10, and use three segments.

In addition to the multisegment generative methods and SCOW, we also evaluate correlation-optimized warping (reviewed in Section 2.2.4), dynamic time warping (Section 2.2.3), and linear parametric warping (Section 2.2.2). For COW, we run the algorithm several times on each trial, varying the slack value s from one to five, and the number of segments m from one to ten. The results reported here are for the best values returned.

The results of these experiments are illustrated in Figure 5.6. The top panel shows results without added query distortion, while the bottom one shows them with the distortion. Recall that the top line for each algorithm shows the treatment accuracy. The center line adds the criterion that the average alignment error must be less than or equal to 24 hours, and the bottom line does likewise but sets the threshold to 12 hours. Points that are highlighted with large squares are significantly different than when we use the three-segment generative method, for the same order

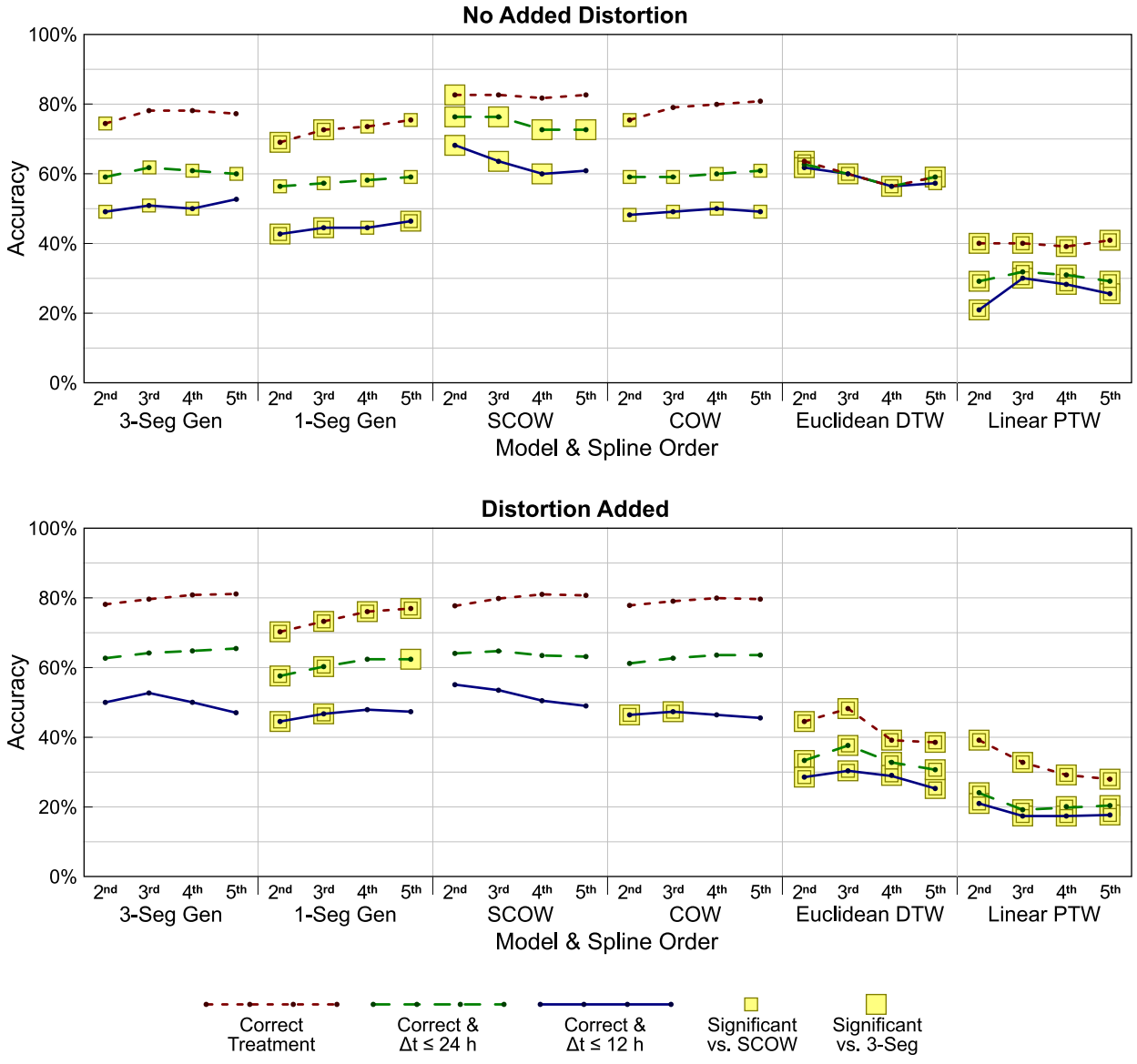

Figure 5.6: Treatment and alignment accuracies of various alignment methods. The first panel represents accuracies when there is no temporal distortion, while the second one shows them when distortion is added. The top lines represent treatment treatment accuracy, while the bottom two lines add the criterion that the predicted times are within 24 and 12 hours respectively of the actual time, on average. Large highlights represent cases in which there is a significant difference in accuracy from the corresponding three-segment generative case ($p \leq 0.05$ with McNemar's χ^2 test), while the smaller highlights show a significant difference from the SCOW results.

of spline (according to McNemar's χ^2 method, with a $p \leq 0.05$.) Points highlighted with small squares are likewise significant versus SCOW.

The class of multisegment algorithms exhibits superior treatment and alignment accuracies. Of these, SCOW is the overall winner. It returns the best results, with high treatment and alignment accuracies. COW and the three-segment generative method are next in terms of accuracy, followed by the one-segment method. Dynamic time warping shows its strong bias toward warping as little as possible, and so its alignment accuracies are nearly the same as its treatment accuracy in the undistorted case. DTW also seems unable to accurately align distorted queries. Finally, the parametric linear case has very low accuracies in all cases.

It should also be remembered that SCOW is much quicker than the three-segment generative method, with its reduced time complexity. For each method, there are a total of 1760 different trials (eleven treatments times ten trials each times four spline orders times four distortions). These 1760 trials require days of computational time using the three-segment generative method, while they can be done in less than two hours using SCOW.

The one-segment model has accuracies almost as good as the three-segment one. This seems paradoxical at first in the undistorted case, since a one-segment model should be sufficient to accurately align all the times. We might expect to see some degradation when using the three-segment model, as it is allowed much more freedom in where it places its segments. One explanation for this result is that the spline interpolation does not create perfect reconstructions of the missing data, and the more expressive three-segment model is better at compensating for this error.

5.2.4 Effects of Stretching and Amplitude Components

We conduct further experiments to evaluate the importance of the stretching and amplitude components of our methods. First, we conduct an experiment in which we effectively remove the amplitude component of our multisegment generative model by fixing the value of a_i to 1.0 for all segments. With all of the probability mass on this single value, the $\log P_a(a_i)$ term in Equation 5.11 becomes zero. In a separate experiment, we set $\sigma_a = \infty$, which makes all amplitude

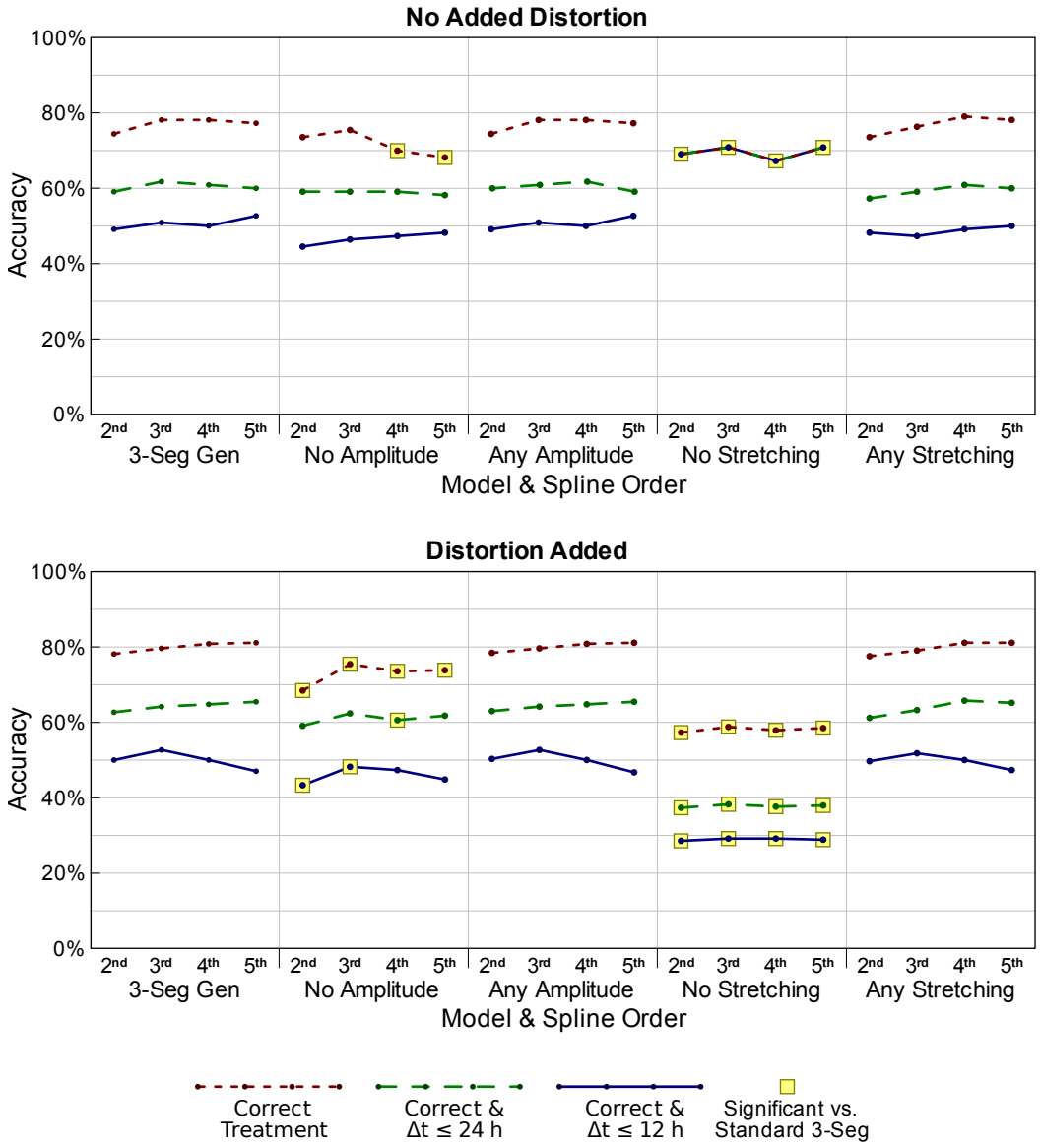

Figure 5.7: Treatment and alignment accuracies when we have removed components of the multisegment generative model. The first method is the three-segment generative model as before. The second disallows any amplitude changes at all, while the third allows any amplitude coefficient with no penalty to the score. Likewise, the fourth disallows stretching and the fifth allows any stretching without penalty. Highlights indicate a significant difference from the unaltered three-segment model ($p \leq 0.05$ with McNemar's χ^2 test).

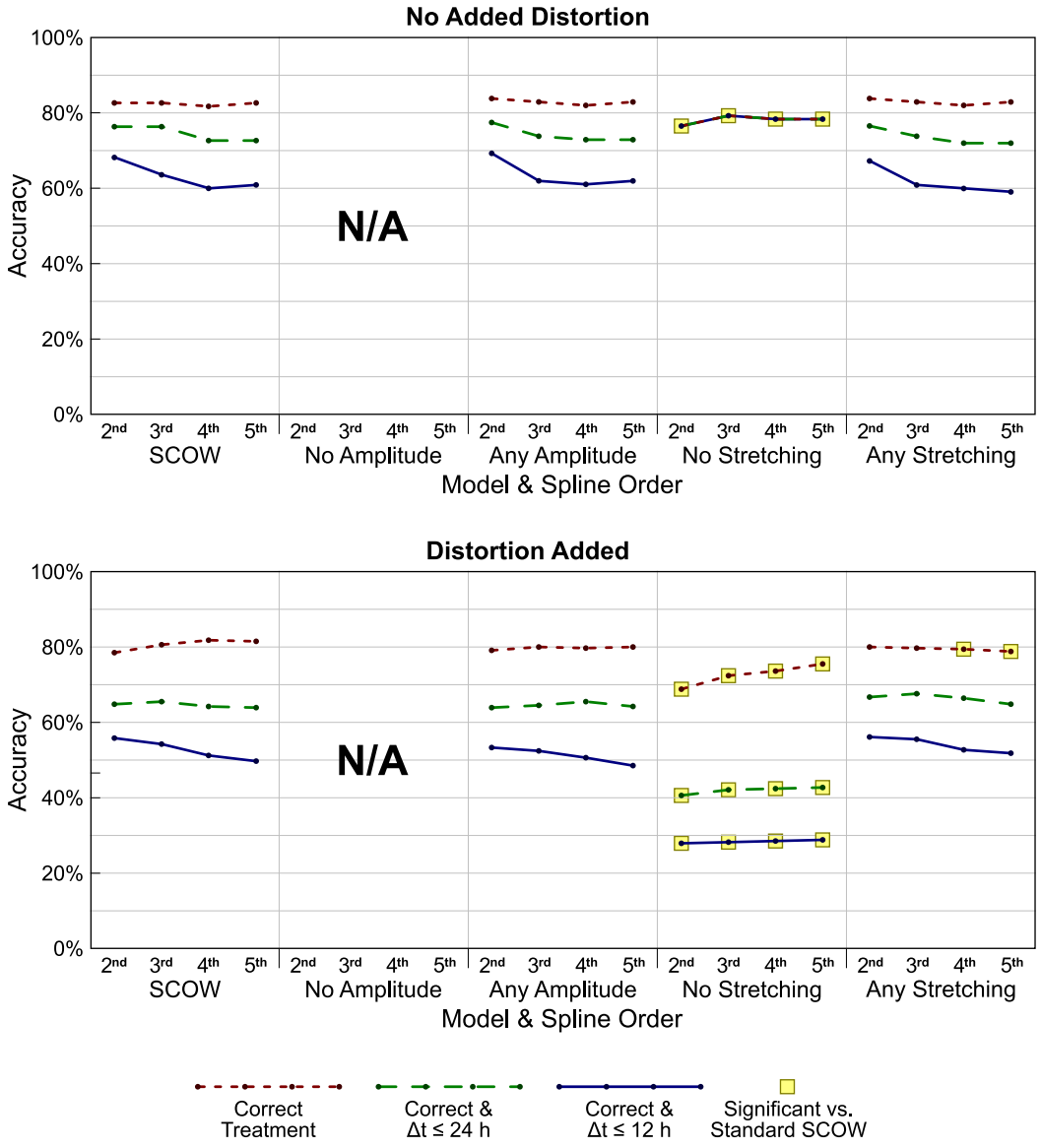

Figure 5.8: Treatment and alignment accuracies when we have removed components of SCOW. The first method is SCOW as before. The third column shows SCOW without an amplitude penalty, the fourth shows SCOW disallowing stretching, and the fifth shows SCOW allowing any stretching without penalty. Disallowing amplitude changes like we did for the multisegment generative model is not applicable to SCOW, because SCOW uses correlation to compare segments. Highlights indicate a significant difference from unaltered SCOW ($p \leq 0.05$ with McNemar's χ^2 test).

changes equally likely. Similarly, we perform experiments in which we force s_i to 1.0 and set $\sigma_s = \infty$. The results of these experiments are shown in Figure 5.7.

Totally disallowing either stretching or amplitude changes has an overall deleterious effect on the accuracy of the alignments. However there seems to be little negative effect in allowing stretching and amplitude changes but not penalizing for greater values. In fact, for the case in which there is distortion in the time series, we see a small advantage in accuracy when there is no stretching penalty. These results imply that the stretching and amplitude components of the model are valuable, but that the accuracy of the alignments is relatively insensitive to the actual penalties selected.

We also perform these experiments for SCOW, as shown in Figure 5.8. However in this case, it does not make sense to do an experiment in which we fix σ_a to 1.0, as we did before. SCOW relies on correlation rather than the Guassian probabilities of the multisegment generative model. Fixing σ_a would only affect the amplitude term in Equation 5.21, and not the correlation term. Forcing σ_a to a single value would have the same effect as ignoring that term altogether, which we have already done. We include the column in Figure 5.8 for symmetry with Figure 5.7.

For the SCOW experiments, ignoring the amplitude component of the method has barely any effect at all on accuracies. It would seem that for this data set, differences in amplitude are not that important to SCOW. By contrast, altering the stretching component has a large effect. Totally disallowing it is deleterious, and is testament to the fact that warping is an important part of similarity searches. Allowing any stretching without penalty has more subtle effects, but still seems to hurt the accuracy of the method. This is especially evident under the strictest correctness criteria.

5.2.5 Effects of Query Size and Number of Segments

We next consider a set of experiments in which we assess the accuracy of computed alignments as a function of both the amount of data in the query, and the number of segments that we allow the alignment method. We are curious how the query size affects the alignment accuracy. We would also like to know if alignment accuracy degrades as our methods are allowed to use more segments than the optimal alignment would require.

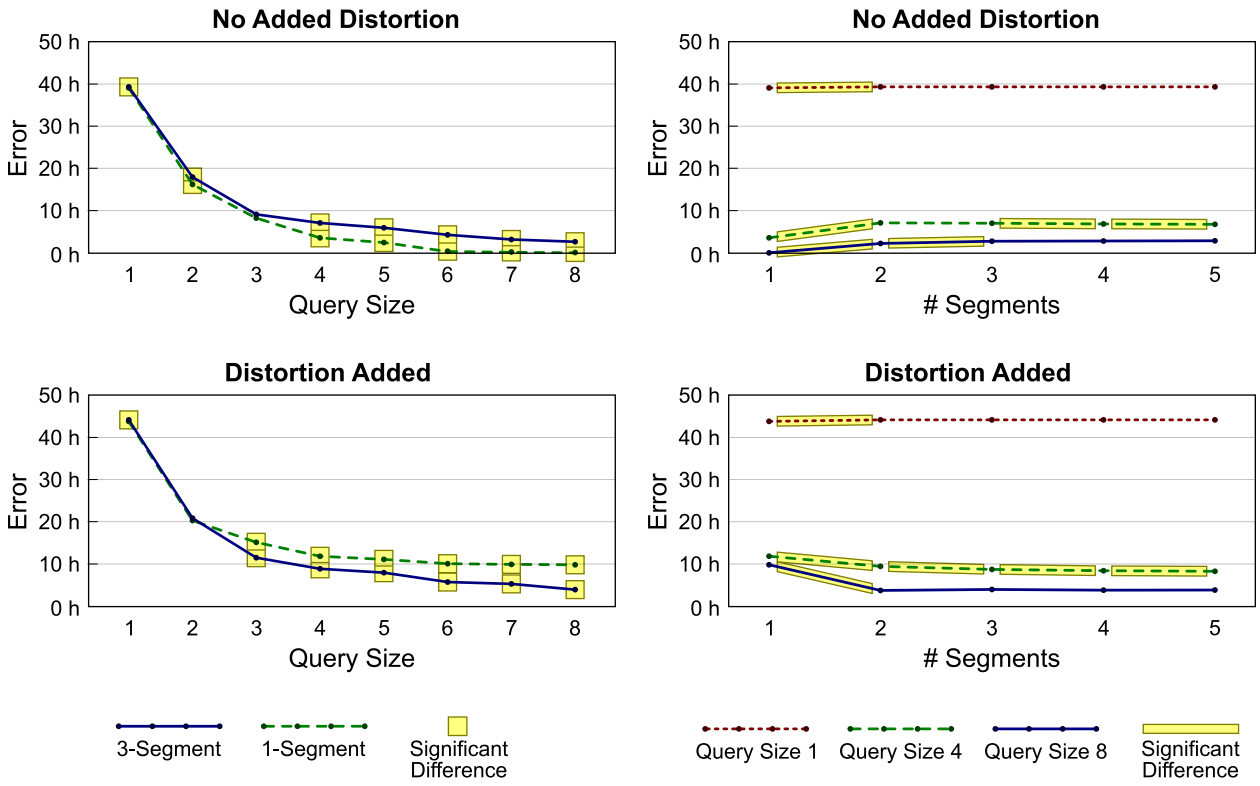

Figure 5.9: Multisegment generative method average alignment error of $1\ {}^{\mu g}/_{kg}$TCDD queries by query size and number of segments. The lines in the left panels show different numbers of segments used by the method, and highlights show cases where this has a significant difference ($p \leq 0.05$ with a two-tailed Student's t test). The lines in the right panels represent different query sizes, and highlighted lines show where increasing the number of segments used makes a significant difference.

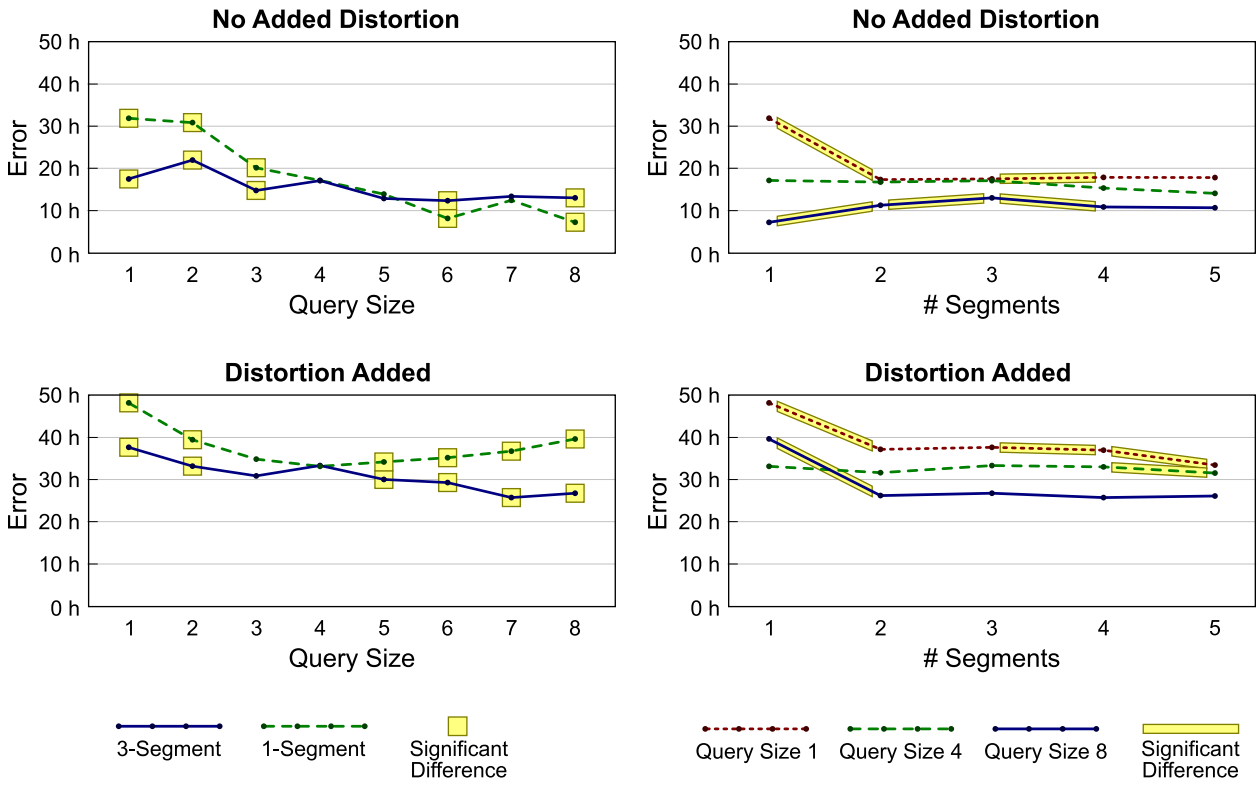

Figure 5.10: SCOW average alignment error of $1\ {}^{\mu g}/_{kg}$TCDD queries by query size and number of segments. The lines in the left panels show different numbers of segments used by the method, and highlights show cases where this has a significant difference ($p \leq 0.05$ with a two-tailed Student's t test). The lines in the right panels represent different query sizes, and highlighted lines show where increasing the number of segments used makes a significant difference.

We start with the multisegment generative algorithm. We restrict our experiments to a single treatment, that being 41 observations of $1\ {}^{\mu g}/_{kg}$ 2,3,7,8-tetrachlorodibenzo-*p*-dioxin (TCDD) taken at eight time points. However, other treatments yield qualitatively similar results. We run several trials, each of which is given a predetermined number of segments m, and query size n. We randomly pick out n observations from different times in the treatment to form each query $\vec{q}$. We use all the remaining observations in the treatment as the database series $\vec{d}$. We interpolate both $\vec{q}$ and $\vec{d}$ every four hours using third-order smoothing B-splines (other orders yield similar results). At last, we compute the best alignment using the m-segment generative method. We do this 100 times for each pair (m, n), and then repeat the experiment under distorted conditions (as defined in Section 5.2.3.) Note that the same 100 random queries are selected for any particular value of n.

We expect the alignment error to generally decrease as we increase the query size. We also expect the one-segment method to perform slightly better when there is no distortion, and the three-segment method to be preferable when there is. However this latter behavior could be confounded for small query sizes, where the three-segment method may not have enough data to determine the segment parameters.

The results of these experiments are shown in Figure 5.9. In the left panels we graph the query size versus the average error, and each line represents a different value for m. The highlighted points are those in which the models performed significantly different than each other ($p \leq 0.05$ by a two-tailed Student's t-test). In the right panels we graph the number of segments versus the average error, and each line is for a different query size n. Adjacent points on the same line thus represent the same query size, but a number of segments that is different by 1. Highlighted line segments thus show cases in which there is a significant difference between the m-segment model and the $(m + 1)$-segment model.

For queries of size two or less, the one-segment model performs slightly better. Its average error is less than that of the three-segment model, by less than one hour. However as the query size grows larger, the expected results become more apparent. When there is no distortion, the one-segment model is adequate. When there is distortion, a multisegment model is clearly preferable. Significantly, the accuracy of the multisegment method is quite robust when it is allowed to use

more segments than necessary. This is important, as in practice we will not generally know the correct number of segments in order to find the best alignment of a query and its best matching series in the database.

We then perform the same experiments for SCOW. The results are shown in Figure 5.10. Although SCOW may align better for very small queries, for medium and large queries its average alignment error is much greater than that for the multisegment generative method. Part of the reason for this is likely because it is a heuristic method, and is not guaranteed to return the best alignment between the two series. However, like the multisegment generative alignment method, SCOW also seems to be robust when allowed more segments than it needs.

5.3 Summary

In this chapter we have detailed two novel segment-based methods that we use to align and compare time series:

- *Multisegment generative*, which performs a complete search to find the best alignment between two series, is based on a generative scoring function.
- *SCOW*, which performs a heuristic search to find the best alignment, uses the Pearson cross correlation as the key component of its scoring scheme.

We have also developed a technique which distorts queries in order to better estimate treatment accuracy for similarity searches, in which there will not be an exact match to the query in the database. We then performed several experiments to show the value of our methods. Both perform well when compared against the previous state-of-the-art methods. However SCOW is much faster than the multisegment generative method, and is more accurate in finding closer treatments in our toxicogenomic data set.

It is significant that our methods align and find nearest treatments more accurately than dynamic time warping, on which much previous work has been done (Keogh and Pazzani, 2000; Aach and Church, 2001; Keogh, 2002). We believe there are two main reasons for this. First, unlike the segment-based methods, the warping path returned by DTW can approximate any arbitrary path

that monotonically increases in both dimensions from the origin to the end point. We believe that this class of alignments is too expressive for our data set—due to its temporal sparsity—which can result in poor alignments. Second, DTW makes an independence assumption about adjacent time points that does not hold for our data. Observations at subsequent times are not independent of one another, and the segment-based methods capture this dependency by scoring all the times within a segment as a unit.

Finally, this work represents the first case in which segment-based methods have been used for similarity queries on gene-expression time-series data. These methods yield good results, and should be studied further.

Chapter 6

Efficient Search for Multisegment Time Series Alignment

One of the main drawbacks of the multisegment generative algorithm is that it is slow. As stated in Chapter 5, its time complexity is $O(n^5)$ where n is the length of one of the series. It takes $O(n)$ time to calculate the score of a single segment. The score of every possible segment must be calculated, and there are $O(n^4)$ of these.

For reasonable sized series ($n \approx 20$) using modern computers, it can take on the order of ten minutes to calculate a single alignment. When performing a similarity search, this alignment must be done for every series in the database. The long time taken is not suitable for our goal of creating an interactive, online tool. Obviously, one solution is massive parallelization. Each alignment is independent of all the others, and so can be "farmed out" to a different processor. However this can be become prohibitively expensive for large numbers of simultaneous queries to large databases. Even with a very large number of processors to do each alignment separately, the amount of time taken to answer a given query is still too long.

In this chapter we attack the problem from an algorithmic point of view, creating heuristics that quickly return a near-optimal alignment. We find that we are able to return answers much more quickly, with little or no loss in accuracy. Parts of the material covered in this chapter were originally published in Smith and Craven (2008).

6.1 The Cone Filter Heuristic

The alignment methods we use work by filling in an alignment matrix Γ. One well studied heuristic in similar time-series alignment problems is to restrict the cells of the matrix that are

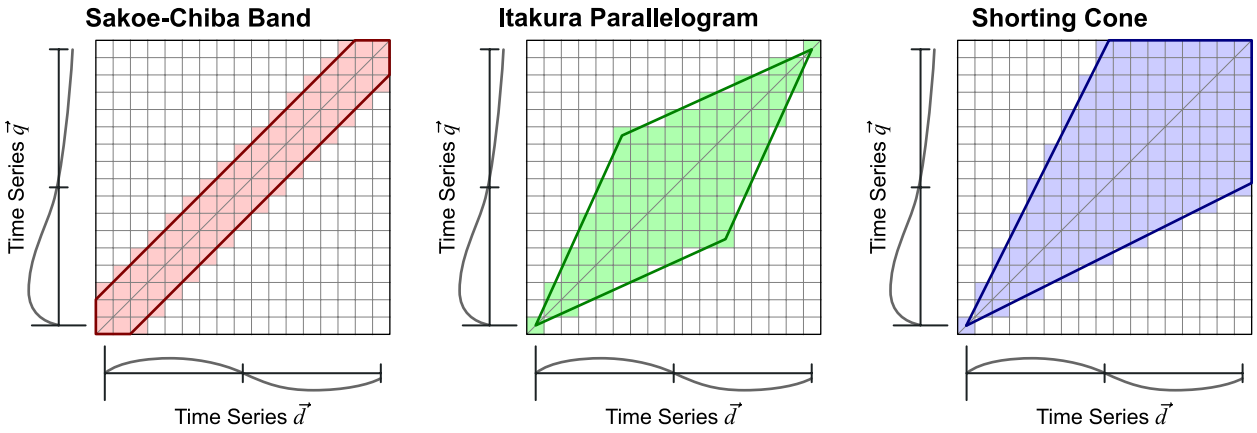

Figure 6.1: Restricting the search in alignment space by shape. Both the Sakoe-Chiba Band and the Itakura Parallelogram are intended for aligning whole series to each other. By contrast, our cone is designed for local alignments in which one series might be shorted.

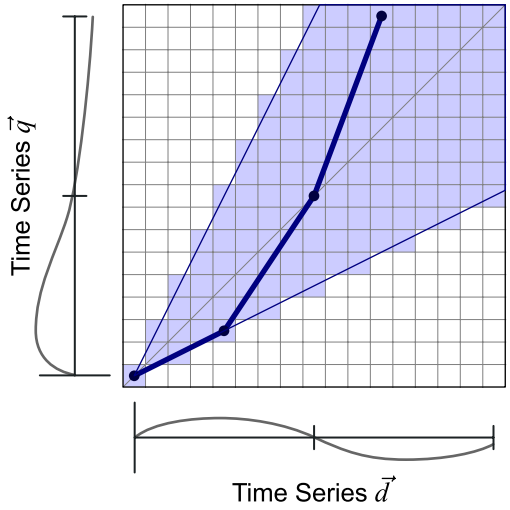

Figure 6.2: Alignment space diagram of the cone filter heuristic for the multisegment generative method. Segments lying outside the cone are not considered during the search for the best alignment path.

calculated. Several ways of doing this are illustrated in Figure 6.1. Each panel shows the alignment space when warping one series against another, and the shaded elements indicate the area to which the search is restricted. The so-called *Sakoe-Chiba Band* (Sakoe and Chiba, 1978) and *Itakura Parallelogram* (Itakura, 1975) are the most commonly used heuristics. The former restricts the

search to a constant distance from the diagonal, while the latter allows progressively more warping closer to the middle. However both of these methods implicitly assume that the alignment being sought is a global one, in which there is no shorting of either input series. Here we consider an approach that we developed, which confines the search to a cone starting at the origin and centered on the diagonal. This is illustrated in the right panel of the figure. We refer to this as the *cone filter heuristic*, since it filters out any segments that do not lie within the cone's area.

Formally, we define the cone used by the heuristic as a slope $c > 1$. We modify Equation 5.14 so that the value of $\gamma(i, x, y)$ is $-\infty$ if it falls outside of the cone:

$$\gamma(i,x,y) = \max_{a<x,\ b<y} \begin{cases} -\infty & \textbf{if } \frac{x}{y} > c \textbf{ or } \frac{y}{x} > c \\ \log P_m(i) + \gamma(i-1,a,b) & \\ \qquad + \text{score}(a,x,b,y) & \textbf{if } x = |\vec{q}| - 1 \textbf{ or } y = |\vec{d}| - 1 \\ \gamma(i-1,a,b) + \text{ score}(a,x,b,y) & \textbf{otherwise} \end{cases} . \quad (6.1)$$

Thus we do not consider any segment with one of its ends anchored outside the cone.

The primary effect of restricting the alignment to a cone is to reduce the number of segments calculated by a constant factor, so we do not expect to see an improvement in its time complexity of $O(n^5)$. Instead we expect to see a constant speedup proportional to the relative size of the cone to the alignment matrix. For a square matrix (i.e. where both series are of the same length), the relative size of the cone depends only on the slope of its bounding rays. With a slope of c, this ratio is $\frac{c-1}{c}$. We expect the execution time of the multisegment method with the cone filter heuristic to take roughly this fraction of the exact calculation's time.

We expect deviation from this value in two cases. First, nonsquare matrices will exhibit smaller ratios, as the cone covers proportionately less of their areas. Second, our calculation assumes that an alignment matrix can be split to an arbitrarily fine granularity. Because the matrix really consists of discrete elements, the ratio of elements covered by a cone will be more than expected for small matrices. For example, in a 5×5 matrix a cone with $c = 2$ will cover 13 cells, for a ratio of $\frac{13}{25}$ rather than $\frac{1}{2}$.

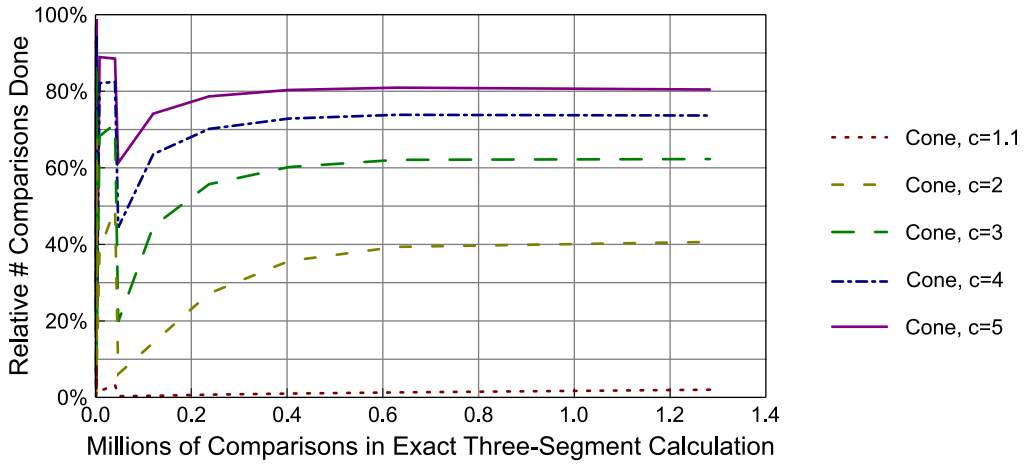

Figure 6.3: Relative speed of the cone filter heuristic, applied to the multisegment generative algorithm. Time is measured by the number of comparisons of a point in each series.

6.1.1 Experiments

Here we evaluate restricting the search in alignment space to a cone, in order to assess (i) its speedup relative to our original multisegment method, and (ii) its ability to find high-scoring alignments and produce accurate time-series similarity searches.

First, we determine how much faster the cone filter heuristic makes our calculations. We measure time in terms of *point comparisons*, or comparisons of a single time point in one series with a single time point in another. This is a good surrogate for calculation time needed. The exact multisegment generative algorithm performs $O(n)$ of these calculations for each of the $O(n^4)$ pairs of segments compared, for a total of $O(n^5)$. For comparison, dynamic time warping performs $O(n^2)$ calculations, while SCOW performs $O(n^3)$.

Figure 6.3 graphs the number of point comparisons done by the exact multisegment generative method versus the fraction done by the cone filter heuristic. Although we obtain good speedup, it is by no more than a constant factor. When the series are of roughly equal length, the time taken is in good agreement with our earlier calculated value of $\frac{c-1}{c}$. We predicted earlier that nonsquare matrices, which have less area covered by the cones, would run faster. These account for the dips

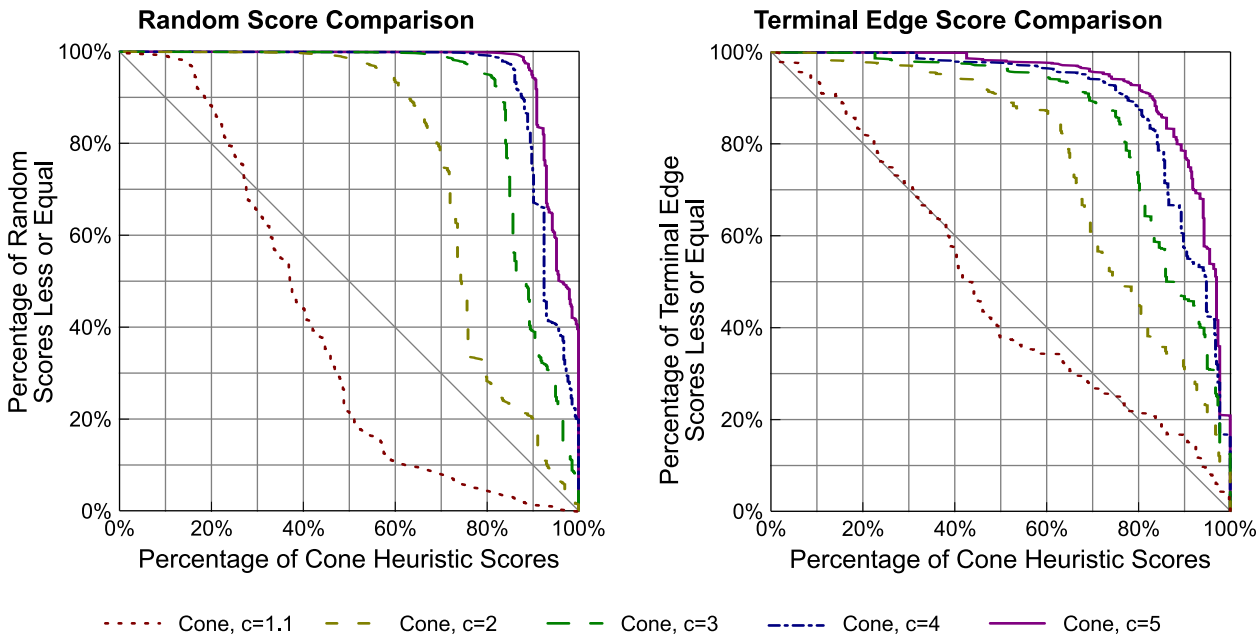

Figure 6.4: Comparison of the cone filter heuristic scores to the scores of the exact multisegment generative method. The left panel compares the cone heuristic scores to the scores of 1000 randomly sampled multisegment alignments. The right panel compares the heuristic scores to the scores on the terminal edges of the alignment matrix of the exact multisegment calculation. These are the best alignments found for each legal shorting of the alignment.

visible in the figure. Additionally as predicted, smaller matrices tend to have larger values on this graph, because of the way cones divide the cells discretely.

Next, we want to make sure that our heuristic is returning reasonable alignments. It is possible that the alignments it is returning are no better than random, and we want to rule that out. We assess the alignments found for each query using the cone filter heuristic by comparing their scores against those of several other alignments.

In the left panel of Figure 6.4, we compare each query score using the heuristic against the scores of 1000 random, legal alignments of the same query. We graph the percentage of heuristic-based scores that are better or equal to the percentage of random-based ones, for the same values of c as before. For example, when $c = 2$, 70% of the cone-based scores are better or equal to at least roughly 80% of the random-based ones. If the cone-based scores were drawn randomly in

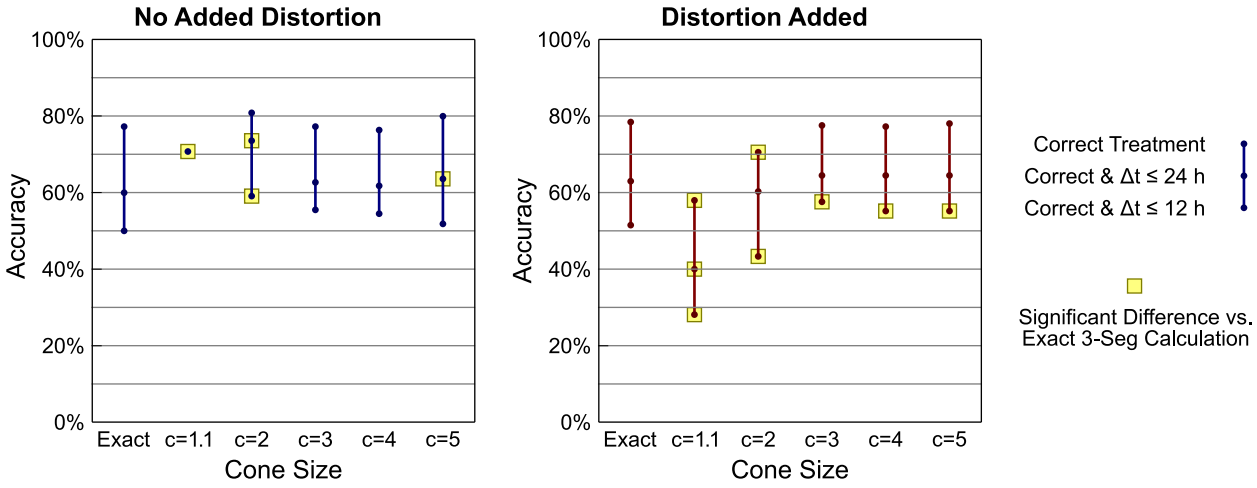

Figure 6.5: Treatment and alignment accuracies for the cone filter heuristic method with varying values of the slope parameter. Also shown for reference are the accuracy values for the exact multisegment method. Highlighted points are significantly different ($p \leq 0.05$) from the exact method by McNemar's χ^2 test.

the same way as the random-based scores, we would expect the curve to roughly coincide with the graphed diagonal (as does the curve for $c = 1.1$). In such a case the area under the curve will be approximately 0.5. On the other hand, if we were to use the exact multisegment method, the area under the curve would be 1.0, since the exact method returns the highest-scoring alignment possible. In this case 100% of exact method scores are better than 100% of random-based scores of the same query. The scores found by our cone filter heuristic result in curves intermediate to these two extremes.

The right panel repeats this experiment, but comparing the heuristic-derived scores with a different non-random set of other scores. Here we compare each query score using the scores in the terminal edges of the alignment matrix for the exact multisegment calculation. These scores include the best one for each possible legal shorting of the alignment. The results are similar to those of the previous test. With $c = 1.1$, the alignment found will likely not be better than picking one of the edge alignments. However with larger cones ($c \geq 2$), there is a good chance that the heuristic will lead to an alignment that scores well.

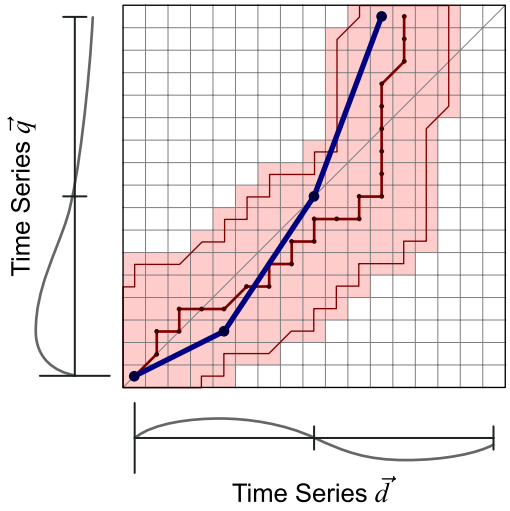

Figure 6.6: Alignment space diagram of the hybrid DTW filter heuristic for the multisegment generative method. We first find an alignment path using our hybrid DTW method, and then we restrict the multisegment search to within a spread s of this path. Here s is two.

Finally we perform the query similarity search/alignment task as in Chapter 5, except using the cone filter heuristic with the multisegment method. For simplicity, we restrict our attention to using only third order (i.e. quadratic) smoothing splines to interpolate. (We choose third order splines because they performed well in the experiments reviewed in Chapter 5.) The results are shown in Figure 6.5. Except for the most narrow cones considered ($c = 1.1$), there is not a loss in accuracy due to finding suboptimal alignments. There may be some benefit to accuracy in using a moderate cone, with $c = 2$ or $c = 3$. If so, this is because such cones preclude the multisegment method from using extreme alignments, such as mapping the beginning of one series to the end of the other or using too great a slope in alignment space.

6.2 The Hybrid Dynamic Time Warping Filter Heuristic

Now we consider an alternative method for speeding up our multisegment alignment method. Here we restrict the seach space by doing a first pass with a method similar to dynamic time

warping, and then considering only multisegment alignments that are close to the found DTW path in alignment space. This method is illustrated in Figure 6.6. This is the *hybrid DTW filter heuristic*.

We refer to the first pass method as *hybrid DTW*, because it combines the dynamic programming of dynamic time warping with a scoring function similar to that used in our multisegment algorithm. The scoring function we use for it is:

$$D(q_x, d_y) = D_E^2(q_x, \alpha d_y) + D_E^2(\frac{1}{\alpha} q_x, d_y) - D_E^2(q_x, \mu_{DB,y}) - D_E^2(\mu_{DB,x}, d_y) \qquad (6.2)$$

where D_E is the Euclidean distance, α is a value chosen by least squares to minimize the first two terms, and $\mu_{DB,x}$ is the average value in the database for time x. Unlike classic DTW, any element $\gamma(x, y)$ can either add to or subtract from the final score. In DTW, the final score is a sum (a normalized sum in our implementation), so it has a strong bias to reduce the number of elements on its alignment path. Our hybrid DTW is able to avoid this bias, making it more likely to choose a path that deviates from the diagonal in alignment space.

Given the alignment path returned by the hybrid DTW calculation, the second step of our approach restricts the search space of the exact multisegment calculation. We define the *spread* s to be the maximum distance from the hybrid DTW path that we will search. As with the cone filter heuristic, we create the hybrid DTW filter heuristic by modifying Equation 5.14;

$$\gamma(i, x, y) = \max_{a<x,\ b<y} \begin{cases} -\infty & \textbf{if } |x - x_h| > s \textbf{ or } |y - y_h| > s \\ \log P_m(i) + \gamma(i-1, a, b) & \\ \qquad + \text{score}(a, x, b, y) & \textbf{if } x = |\vec{q}| - 1 \textbf{ or } y = |\vec{d}| - 1 \\ \gamma(i-1, a, b) + \text{score}(a, x, b, y) & \textbf{otherwise} \end{cases} , \qquad (6.3)$$

for all points (x_h, y_h) in the hybrid DTW path. Thus we only consider segments that both begin and end within s of the hybrid DTW path.

Like classic DTW, the time complexity of hybrid DTW is $O(n^2)$, where n is the length of one of the series being aligned (assuming they have similar lengths). The maximum length of the path it returns is $2n$. This gives us a maximum bound on the number of segments calculated for the

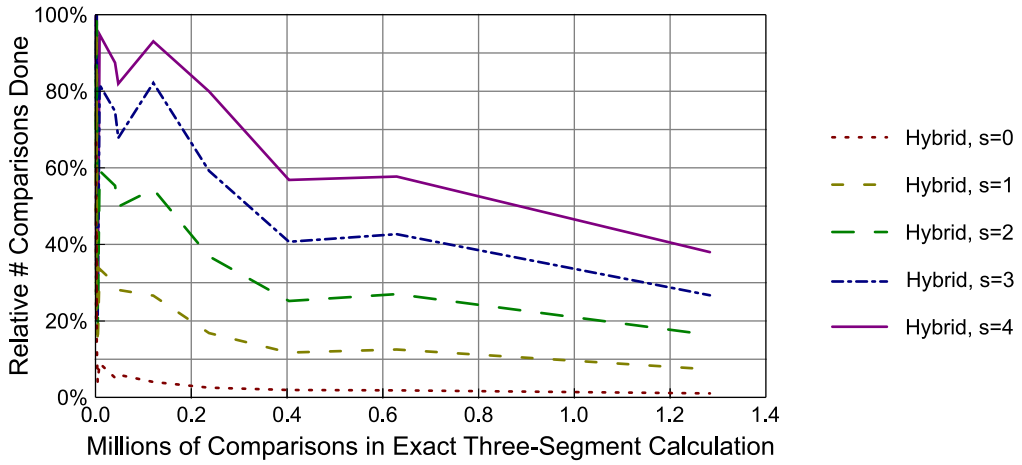

Figure 6.7: Relative speed of the hybrid DTW filter heuristic, applied to the multisegment generative algorithm. Again, time is measured by the number of comparisons of a point in each series.

multisegment method of:

$$\sum_{i}^{2n}(i-1)(2s+1) = \frac{1}{2}((2n)^2-(2n))(2s+1), \tag{6.4}$$

where s is the spread. We multiply this value by the $O(n)$ time required to calculate the score of a segment, and obtain a total time complexity of $O(n^3 s)$ for the hybrid DTW filter heuristic.

6.2.1 Experiments

As with the cone filter heuristic, we assess the hybrid DTW filter heuristic by considering (i) its speedup relative to the original multisegment method, and (ii) the quality of the alignments it finds. We evaluate these criteria with s ranging from zero up to four (which corresponds to a range from zero to sixteen hours in our time series).

Speedup is shown in Figure 6.7, which is again measured in terms of point comparisons. With $s = 0$ or $s = 1$ we obtain speedups of an order of magnitude. We again see the dips corresponding to nonsquare matrices. The best speedups occur for the largest matrix sizes. In contrast to the cone filter heuristic, the ratio of comparisons done still appears to still be decreasing for the largest values measured. This is what we would expect with a better time complexity.

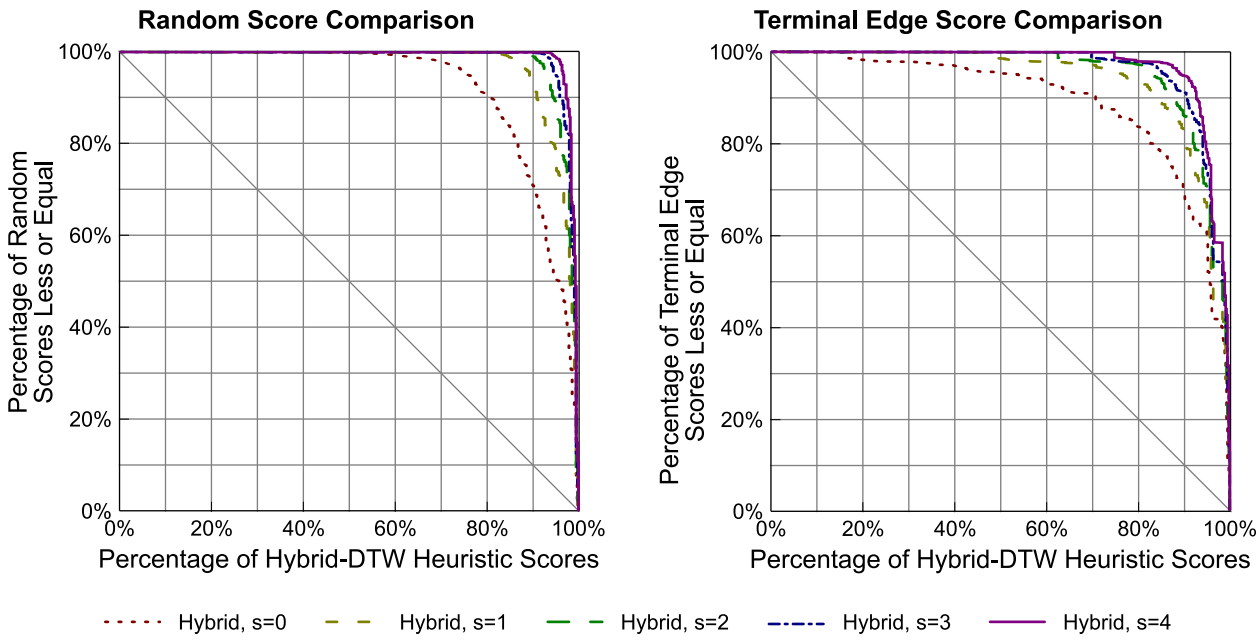

Figure 6.8: Comparison of the hybrid DTW filter heuristic scores to the scores of the exact multisegment generative method. As before, the left panel compares the heuristic scores to the scores of 1000 randomly sampled multisegment alignments, while the right panel compares the heuristic scores to the best terminal-edge scores under the multisegment generative method.

We look at the alignment scores, and as before we compare them to the scores of both random alignments and terminal-edge alignments. The results of these score comparisons are shown in Figure 6.8. The heuristic does well here, even when s is zero. Most of the scores found using this heuristic are better or equal than the edge and random scores for the same query.

Finally, Figure 6.9 shows the treatment and alignment accuracies for the exact multisegment method and the method using the hybrid DTW filter heuristic. There is no significant difference in accuracy when using the heuristic versus doing the exact multisegment calculation. For completeness, the figure also shows the accuracies when using the hybrid DTW method by itself. Although it is more robust to distortion than ordinary DTW and seems to align well, its accuracy (especially treatment accuracy) is still significantly worse than that of the multisegment method. Note that

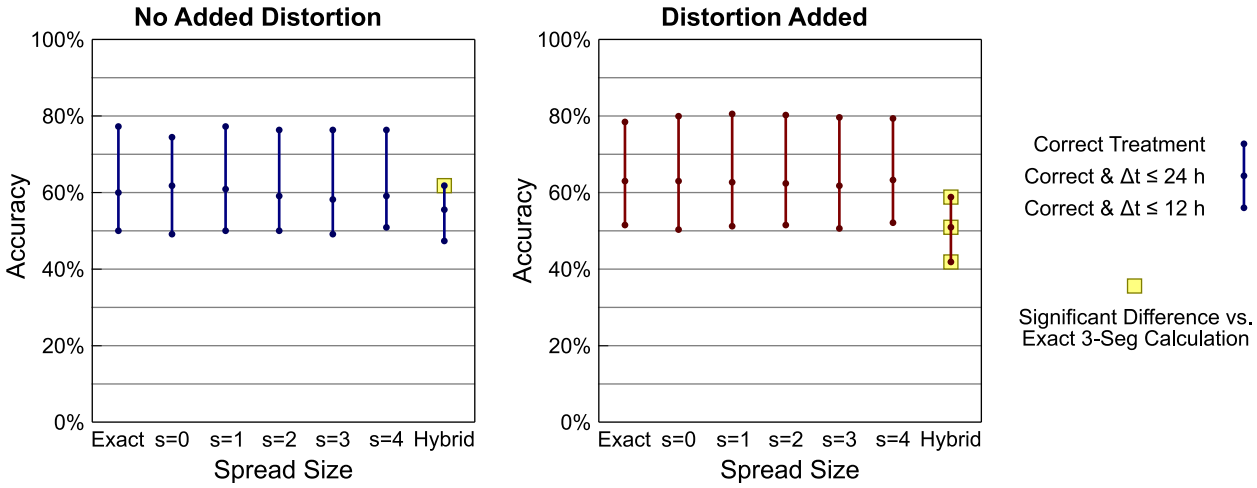

Figure 6.9: Treatment and alignment accuracies for the hybrid DTW filter heuristic with varying values of the spread parameter. Also shown for reference are the accuracy values for the exact multisegment generative method and the hybrid DTW method on its own.

this method is not simply the hybrid DTW filter heuristic with $s = 0$, which returns a segment-based alignment. The alignment that the hybrid DTW method returns is more like that returned by traditional DTW (i.e. the jagged alignment path in Figure 6.6).

Taken together, these results imply that the hybrid DTW's alignment paths are a good approximation to those found by the multisegment method. Using spread values of zero or one with the filter heuristic has the potential to speed up the calculation greatly while not significantly harming the accuracy provided.

6.3 The Alternating Search (SCOW) Heuristic

Finally, we consider changing the search heuristic so that we search for knots (segment endpoints) with respect to each dimension in alternation, rather than searching for them all at once. This is the same heuristic that SCOW uses, as was detailed in Section 5.1.2. However, here we use the scoring function of the multisegment generative method. This allows us to evaluate the effectiveness of the search heuristic itself. The search heuristic is defined formally in Equations 5.17

through 5.21, but the score function we use here is as defined in Equation 5.13 rather than Equation 5.22.

To review, we start by doing two searches. In one, we search for the highest scoring segments whose knots are evenly distributed with respect to $\vec{q}$ but free to vary with respect to $\vec{d}$. For the other search, we switch the series in which the knots can vary. We take the search that results in the highest scoring segments, and use that to begin the alternating search. In each step, we swap which dimension has constant knots and which can vary. We use the best knots found by the previous step as the locations of the knots in the constant dimensions. We keep alternating this way until the search converges. The alignment path so found is not guaranteed to be the same one as in the exact algorithm, but is expected to be near-optimal.

Recall that each iteration must score at least as well as the previous one, and that we only consider knots from a limited number of locations. Because of this, the algorithm is guaranteed to converge. However, as with SCOW, the algorithm forces us to use the maximum number of segments. This is in contrast to the exact multisegment generative method (including under the previous heuristics in this chapter), which can return a smaller number of segments if doing so scores better.

Like SCOW, the time complexity of this heuristic is $O(imn^3)$. Again, i is the number of iterations performed, m is the number of segments, and n is the length of one of the interpolated series to be aligned. Each segment calculation can be calculated in $O(n)$ time. The number of predecessors for any one element of one of the matrices is also $O(n)$. The size of a matrix is $O(mn)$. With i iterations, that gives the total of $O(imn^3)$. However, i and m are usually small numbers, and so the complexity is dominated by n^3. Based on the speedup of SCOW, we expect the alternating search heuristic to execute much more quickly than the exact multisegment calculation.

6.3.1 Experiments

We perform the same tests as with the other two heuristics to assess the speedup and alignment quality when using the alternating search heuristic.

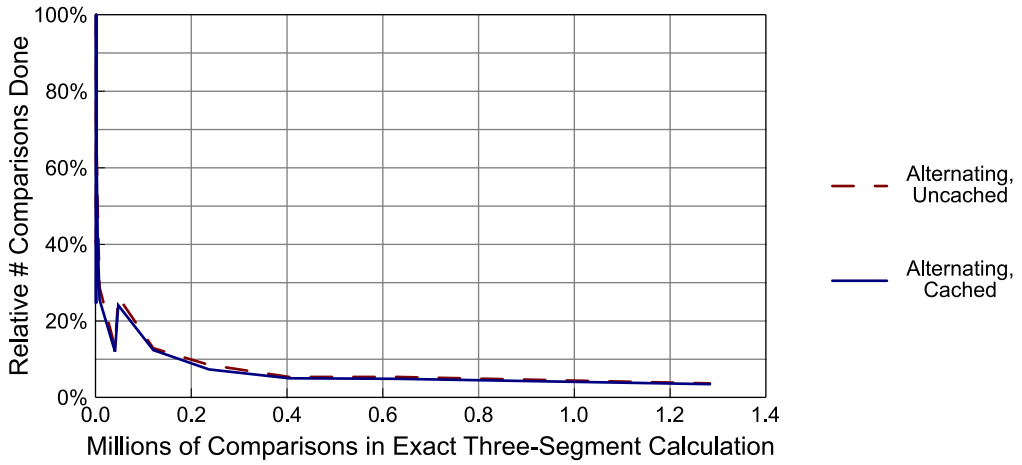

Figure 6.10: Relative speed of the alternating search heuristic, applied to the multisegment generative algorithm. As before, time is measured by the number of comparisons of a point in each series. The cached calculation stores the score of each individual segment, so that we do not have to recalculate it in successive iterations.

We start by showing the speedup in Figure 6.10. The dashed line shows the speedup using the algorithm as we have explained it. The solid line beneath it uses caching to reduce the calculations necessary. Searches often recalculate the scores of segments that have already been seen in previous iterations. By storing these values, we can reduce the time taken by the algorithm. In fact, it is possible for the uncached version to require more calculations than the exact multisegment generative method, as we rescore the same segments during successive iterations of the algorithm. However, this usually only happens for very small matrix sizes. For some larger ones, we obtain a speedup of more than 30 times. Based on this figure, we conclude that the alternating search heuristic is indeed able to speed up the calculation greatly.

Next we look at the alignment scores found, comparing them to our random alignment scores and terminal-edge scores. These results are shown in Figure 6.11. This heuristic does extremely well here, with the areas under the curves approaching 1.0. The hybrid DTW heuristic performs similarly when the spread s is large enough, but in these cases its speed is not nearly as fast as that of the alternating search heuristic.

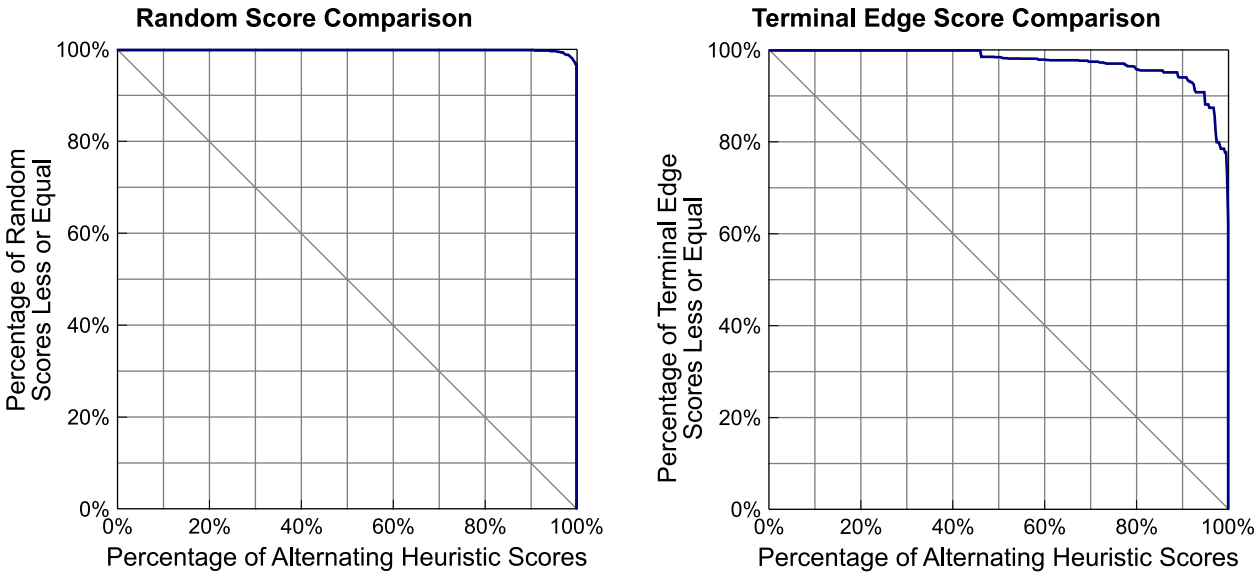

Figure 6.11: Comparison of the alternating search heuristic scores to the scores of the exact multisegment generative method. Again, the left panel compares the heuristic scores to the scores of 1000 randomly sampled multisegment alignments, while the right panel compares the heuristic scores to the best terminal-edge scores under the multisegment generative method.

Finally we look at the classification and alignment accuracies in Figure 6.12. Here there is no significant difference between using the alternating search heuristic, and not. The smallest p-value for the accuracy differences shown in the figure is 0.317.

We conclude that the alternating search heuristic is an excellent one to use. It cuts the search time drastically, while barely affecting the quality of the alignments found.

6.4 Summary

In this chapter we have looked at three heuristics for speeding up the multisegment generative method:

- The *cone filter heuristic*, which restricts the alignment space searched to a cone around the diagonal.

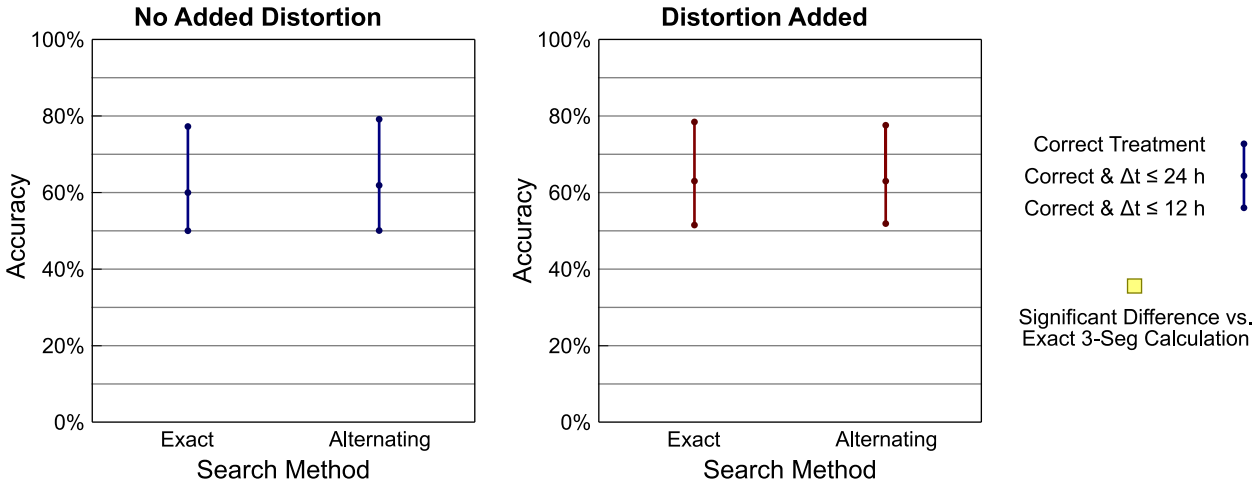

Figure 6.12: Treatment and alignment accuracies for the alternating search heuristic method with varying values of the spread parameter.

- The *hybrid DTW filter heuristic*, which restricts the alignment space to an area around the warping path found by an algorithm similar to dynamic time warping.
- The *alternating search heuristic*, which uses SCOW's technique of searching for knot values in only one dimension at a time.

We have performed experiments to assess how well these methods cut the time needed to align series, and their impact on the quality of the alignments found. The cone filter heuristic shows promise to improve the accuracy of the calculated alignments, since it prevents radical alignments that should not be allowed. However, it does not speed up the alignment process by more than a constant amount. By contrast the hybrid DTW filter and alternating search heuristics both cut the time complexity of the search from $O(n^5)$ to $O(n^3)$ without seriously affecting the accuracy of the alignments found. The alternating search heuristic performs particularly well, speeding up the search by over 30 times.

Finally, these experiments give us additional insight into the success of the SCOW algorithm. Using the alternating search heuristic makes the multisegment generative method much more efficient, but it does not significantly affect its accuracy. We conclude that SCOW's higher accuracy is a result of its scoring function, not the search method it uses.

Chapter 7

Computing Clustered Alignments of Time Series Data

Previous approaches to aligning gene-expression time series have assumed that all genes should be aligned in lockstep with one another. In other words, these methods assume that the transformation that specifies how one series relates to another is the same for all genes. In this chapter, we present a novel approach that finds clusters of genes such that the genes within a cluster share a common alignment, but each cluster is aligned independently of the others. Our method is similar to *k*-means clustering (Duda et al., 2000) in that it alternates between assigning genes to clusters and recomputing the alignment for each cluster using the genes assigned to it. Much of the material covered in this chapter was originally published in Smith et al. (2009).

We start by describing the method, and then detail experiments that demonstrate its utility.

7.1 Clustered Alignments

Figure 7.1 illustrates the motivation for identifying clusters when aligning gene-expression time-series data. Both Treatment B and Treatment C are suitable matches for the query. However, in Treatment B all the genes are aligned to the query's genes in the same way, and in Treatment C they are not. Aligning all the genes together is the usual approach when aligning gene-expression series (Aach and Church, 2001; Bar-Joseph et al., 2003; Criel and Tsiporkova, 2006). This is because warping genes individually is highly prone to error (Bar-Joseph et al., 2003), as observations are sparse data and reconstructions are imperfect. However because there are many dependencies among the genes, many of them will often respond in concert. Thus, by assuming that all genes must be aligned in lockstep, we can average out alignment error that occurs in individual genes.

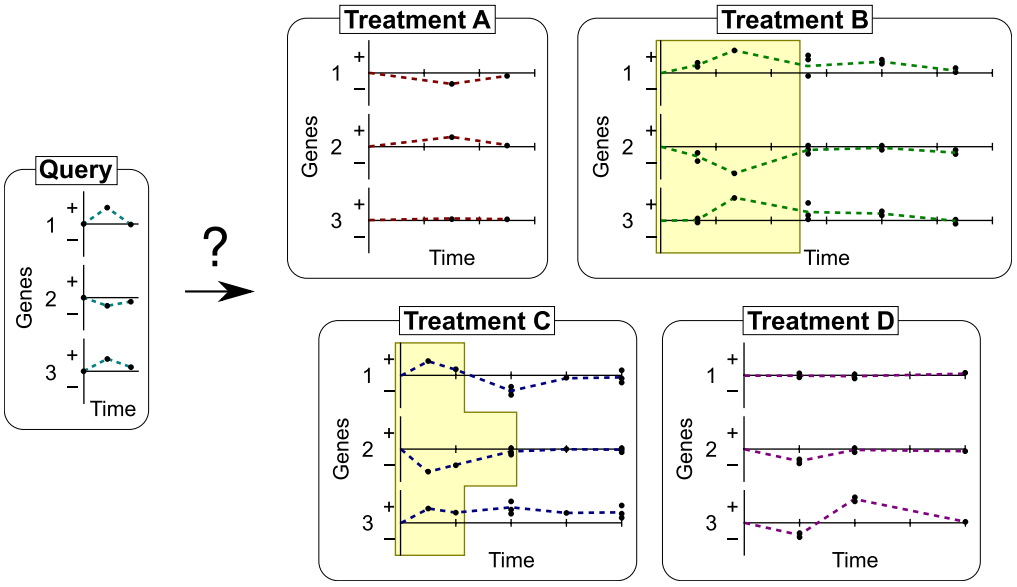

Figure 7.1: Toy gene-expression similarity problem. Both Treatments B and C are similar to the query. However in Treatment B, all the genes are warped as a unit. In Treatment C, there are two "clusters" of genes. The first and third genes are aligned together as one cluster, and the second gene forms the other.

However, we know that subsets of genes often exhibit different responses in two related time series. For example, as the dose of 2,3,7,8-tetrachlorodibenzo-*p*-dioxin is increased, some genes show more exaggerated responses whereas others do not. This is the kind of situation illustrated by Treatment C in Figure 7.1. Here, the first and third genes can be warped together to match the query, but the second gene should be aligned separately.

The clustering algorithm we have developed represents a middle-ground approach to this alignment problem. We do not want to align every gene separately, because that would introduce large error into the alignments found. Nor do we want to assume that we should align all genes together, because this simplification can cause us to miss important distinctions between genes. Rather, our goal is to find sets of genes that would have very similar alignments if they were aligned independently, without error. We can then warp the genes in one of these "clusters" as a unit, while allowing the genes in different clusters to be warped independently.

Table 7.1: Pseudocode for our clustered alignment algorithm.

```
procedure ClusterAlignments(series d⃗, series q⃗, # clusters k):
    // initialize cluster centroids //
    centroid[1] ← null alignment
    for all (genes g):
        possible[g] ←ScoreGene(q⃗, d⃗, g,Align(q⃗, d⃗, {g}))
        best[g] ←ScoreGene(q⃗, d⃗, g, centroid[1])
    for (i ← 2 to k):
        worst ← argmin_g(best[g] − possible[g])
        centroid[i] ←Align(q⃗, d⃗, {worst})
        for all (genes g): best[g] ← max(best[g],ScoreGene(q⃗, d⃗, g, centroid[i]))
    repeat:
        // assignment step //
        for all (centroids c): set[c] ← ∅
        for all (genes g):
            s ← argmax_c(ScoreGene(q⃗, d⃗, g, c))
            set[s] ← set[s] ∪ g
        // update step //
        for all (centroids c): c ←Align(d⃗, q⃗, set[c])
    until sets converge
```

The algorithm we have devised is a variant of traditional *k*-means clustering (Duda et al., 2000). In *k*-means, each cluster is represented by a centroid and the clustering process involves iteratively refining the locations of these centroids. For example, if we were clustering points in $\mathbb{R}^n$, each centroid would be represented by a point in $\mathbb{R}^n$. In our clustered alignment method, each "centroid" is represented by a separate alignment. In our algorithm, as in standard *k*-means, the number of clusters is determined by a parameter k that is provided as an input. We then alternate between assigning genes to clusters based on how each cluster is aligned, and recalculating the alignment of a cluster, until the process converges.

Note that in contrast to previous methods which have focused on identifying clusters of genes that have similar expression profiles, our algorithm is focused on identifying clusters in which

the genes have similar warpings. The genes within one of our clusters may have very different expression profiles.

Table 7.1 shows the pseudocode for our alignment clustering method. It takes as input two series ($\vec{d}$ and $\vec{q}$) and the number of clusters k. It relies on the subroutines *Align*, which returns the best alignment between two series based on a given set of genes, and *ScoreGene*, which returns the score of two series when aligned using a given alignment and a specified gene. We use SCOW (described in Section 5.1.2) to perform these functions, using the *SCOWAlign* function for *Align* while using Equation 5.21 for *ScoreGene*. However, we could substitute any other alignment algorithm for this purpose.

The first step in the method is to assign the initial alignment centroids. We use a greedy method, similar to that used by Ernst et al. (2005) to select a representative set of gene alignments as the centroids. The first centroid is taken to be the null alignment, which represents no warping. For each gene we record a best possible score (when the alignment is based solely on that gene), and the best score seen so far for that gene using one of the current centroids. Each additional centroid is initialized by finding the gene with the largest difference between its best score so far and its possible high score. The new centroid is the alignment calculated using this selected gene alone. After each new centroid is determined, the best scores for all the genes are modified to take the new centroid into account. We proceed until all k centroids are defined.

Next we perform the assignment step and the update step in turn until convergence. For the assignment step, we score every gene with every cluster's centroid and assign the gene to the cluster with the highest score. For the update step, we set each centroid to the alignment calculated by aligning $\vec{q}$ and $\vec{d}$ using just the set of genes assigned to the cluster.

We continue iterating the assignment and update steps until the cluster assignments do not change. Because SCOW performs a heuristic search, however, it is possible that the process will not converge. In practice, this is seldom a problem. We can simply stop iterating after a large number of iterations, or when infinite loop conditions are detected by retaining a short history of cluster assignments. Alternatively, we can guarantee convergence by using an alignment algorithm that is exact.

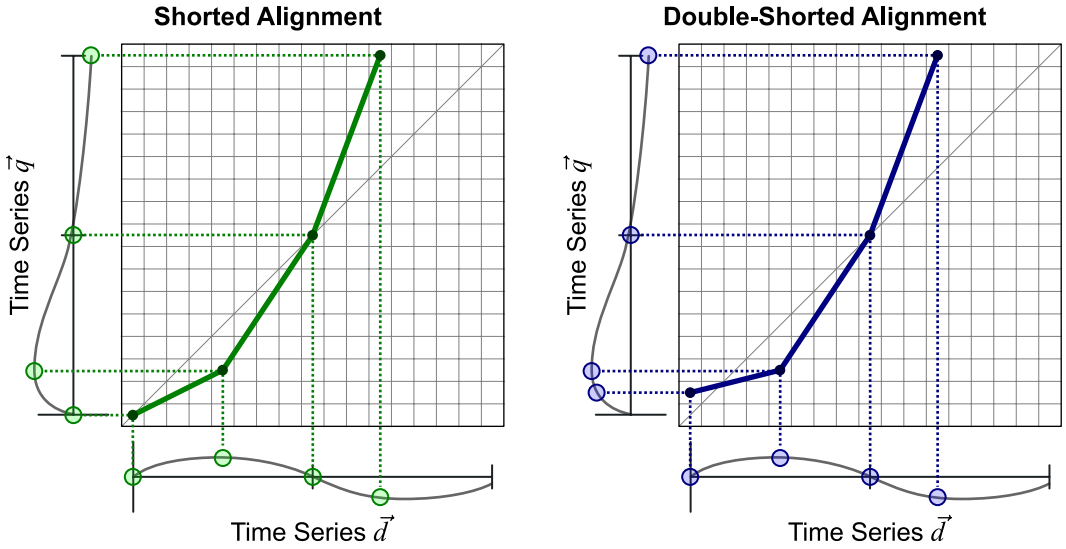

Figure 7.2: Shorted and double-shorted alignments in warp space. In a shorted alignment, the ends of the two series are not aligned. In a double-shorted alignment, both the beginnings and ends are not aligned.

7.2 Double-Shorted Alignments

In addition to the EDGE data set we have previously used, we also test our clustering algorithm on a gene knockout experiment. A gene knockout experiment is one in which one or more genes are disabled in an experimental population, but not a control population. The knockout experiment we consider involves the deactivation of the Mop3 circadian gene in mice. Circadian genes regulate behaviors such as the sleep-wake cycle, and usually have a period of 24 hours. In this experiment, the mice are observed over the course of a day, in order to assess which genes show different expressions when the Mop3 gene is knocked out.

In this knockout experiment, the important differences in the time series do not originate from a particular time at which a treatment has been applied. This contrasts with our toxicology data set, in which there is a well-defined "time zero." Therefore, we cannot assume that the starting time points of the two series should be aligned, and thus we want to allow double-shorted alignments. Figure 7.2 illustrates the crucial difference between the shorted alignments we have considered

in previous chapters, and a double-shorted alignment. In a double-shorted alignment, neither the beginning nor the end of the series can be assumed to be aligned.

There are two problems with using a double-shorted aligning method in this domain that we wish to avoid. These are illustrated in Figure 7.3. First, since the control has a high chance of being cyclic, it can happen that there are two correct, distinct warping paths available. The paths in the figure are in fact continuous, which can be seen by phase-shifting the control series back seven spaces. We want an alignment method that recognizes that these two paths are really different parts of the same path, and is not forced to choose one or the other.

The second possible pitfall is that the alignment method might choose a very short path during which the two series are coincidentally very similar, and so overestimate the similarity of the two series. This is shown in the right panel of Figure 7.3, in which the series are very different except at the beginning of the knockout series and end of the control series. Even when the total similarity score is not an issue (as when comparing only two treatments), this can cause the aligner to miss a longer, imperfect path that might show the true relationship between the series.

We address these problems via a two-step solution. First, we concatenate the control series with itself to create a new 48 hour control series. The control animals have functioning circadian cycles, and so gene expression levels are likely to follow a 24 hour cycle. Thus we concatenate a single day to estimate two day's worth of data. Then, we allow the alignment found to short with respect to the control series, but not with respect to the knockout series. Thus every time in the knockout series will be aligned to some time in the control series, though the converse is not necessarily true. This approach is illustrated in Figure 7.4. Of course, it would be preferable to directly observe two days worth of data from the control group rather than artificially concatenating it. However because of the cost of each separate observation, we currently do not have such data.

We modify SCOW to disallow shorting in the knockout series as follows. We set $\vec{d}$ to be the control series, and $\vec{q}$ to be the knockout series. First, we replace Equation 5.17 with the following two equations:

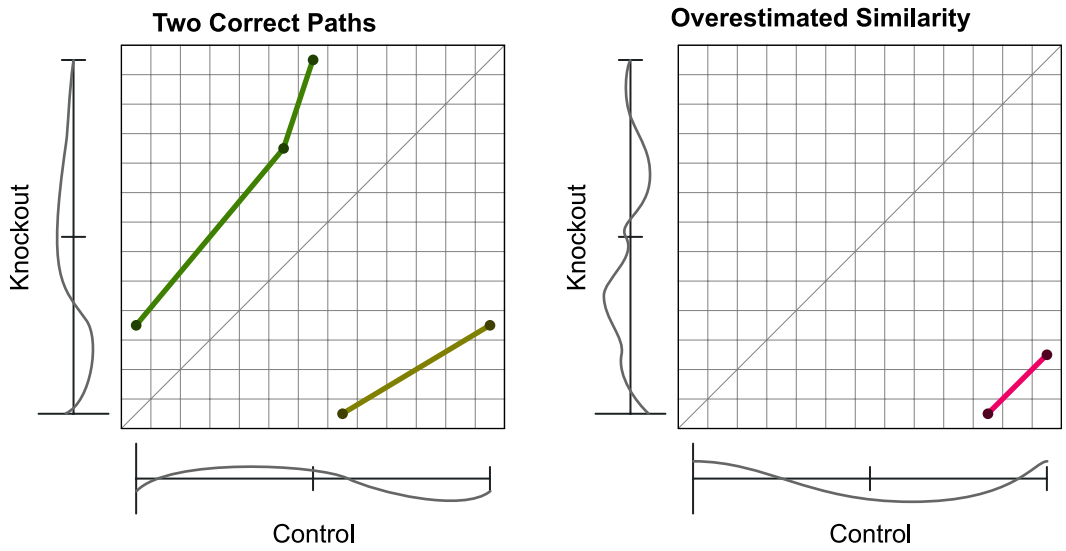

Figure 7.3: Possible pitfalls of double-shorting with the Mop3 gene knockout experiment. In the left panel, two distinct paths align the series well. If the control series is cyclic, the two paths will be continuous with each other, and should chosen together. In the right panel, the two series have very little in common. However, a double-shorted alignment can overestimate their similarity if it compares only a very short region of the two.

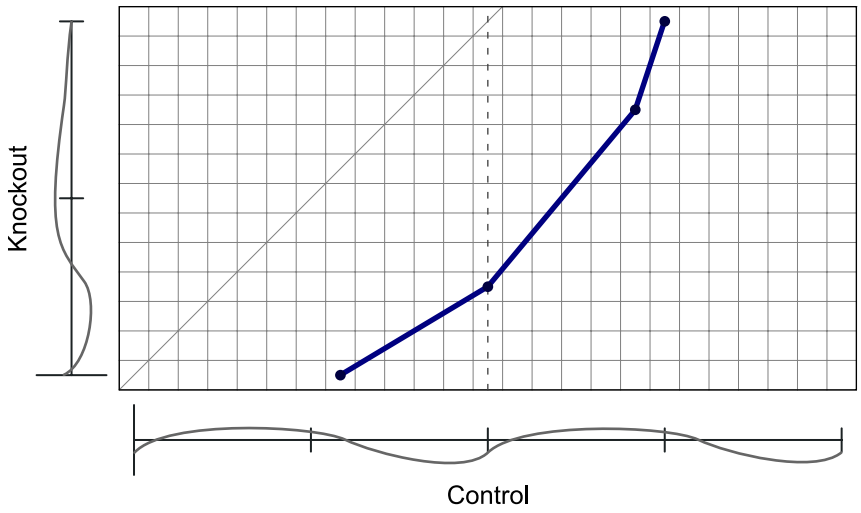

Figure 7.4: Double-shorted segment-based alignment. We solve the short alignment problem by concatening the control series with itself to double its length, and allowing the resultant control series to short but not the knockout series.

$$\gamma^d(0,j) = 0, \tag{7.1}$$

$$\gamma^q(0,j) = \begin{cases} 0 & \textbf{if } j = 0 \\ -\infty & \textbf{otherwise} \end{cases}. \tag{7.2}$$

This allows the best path in Γ^d to start from any point $\gamma^d(0,i)$, but restricts the best path in Γ^q to start at $\gamma^q(0,0)$. Hence $\vec{d}$ can be shorted at the beginning but not $\vec{q}$, and the zero times of the two series need not correspond. The dynamic program continues as previously defined, filling Γ^q and Γ^d using Equations 5.18 and 5.19, respectively. At the end of each iteration, the equation used to find the best alignment path varies, depending on if we are calculating Γ^d or Γ^q:

$$\text{bestscore}(\Gamma^d) = \max_j \gamma^d(m,j), \tag{7.3}$$

$$\text{bestscore}(\Gamma^q) = \gamma^q(m, |\vec{q}| - 1). \tag{7.4}$$

As before, this ensures that the best path found in Γ^d can end at any point $\gamma^d(m,j)$ (recall that m is the number of segments), but the best path in Γ^q must end at the final element, $\gamma^q(m, |\vec{q}|-1)$. Using this modified version of SCOW, we can obtain an alignment like the one pictured in Figure 7.4.

7.3 Experiments

We are interested in testing the ability of our clustered alignment algorithm to identify sets of genes that should share a common alignment. We first conduct an experiment designed to determine if our clustered alignment method is able to find more accurate alignments when there are sets of genes that have different, known "correct" alignments.

This experiment is largely similar to those detailed in previous sections, and we use the EDGE toxicology data set. We pick out a small number of random observations from the same treatment on which to perform each trial, using the remaining treatments and observations as the database.

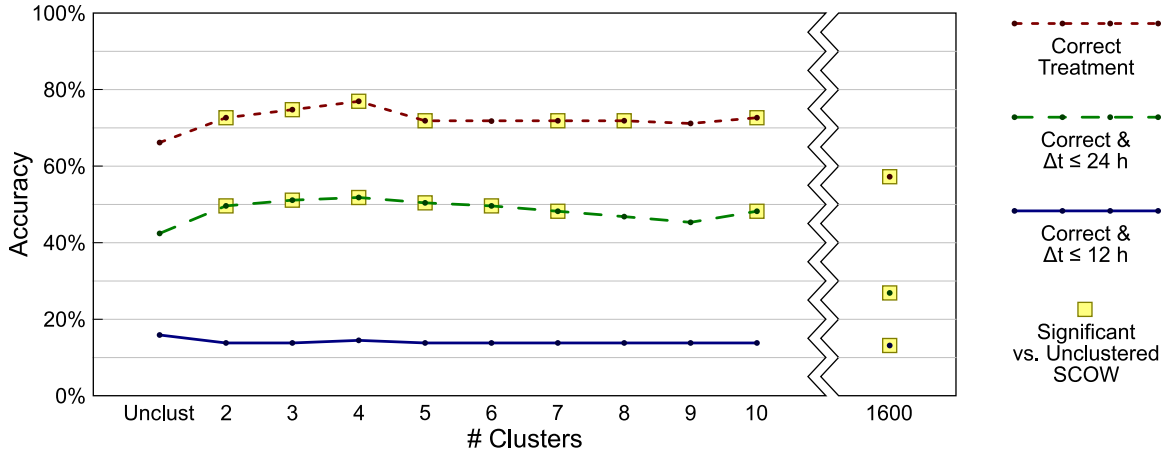

Figure 7.5: Treatment and alignment accuracies, varying by the number of clusters when using SCOW. In the final case (1600) we warp every gene separately. Highlighted points are significantly different from the unclustered case, ($p \leq 0.05$ using McNemar's χ^2 test).

We interpolate this test set and each treatment within remaining within the database using third-order smoothing splines, reconstructing the query series $\vec{q}$ and eleven different database series $\vec{d}$. We then align $\vec{q}$ against each $\vec{d}$, and classify $\vec{q}$ to be the same as the series with which it has the highest scoring alignment.

The difference from previous experiments is that we simultaneously apply five different temporal distortions to every query: each one is applied to $^1/_5$ of the genes. We then run our clustered alignment method, in conjunction with SCOW, on the data, allowing the number of clusters k to range from one (i.e. unclustered, ordinary SCOW) to ten. We also run the experiment with $k = 1600$, which warps every gene separately.

The results for queries containing three or more observations are shown in Figure 7.5. These results show the value of the clustered alignment approach with this data set. The accuracy of the alignments increases as k increases, until $k = 4$. After this point, there is a slight degradation in accuracy. For all values of k tested, however, the treatment and the 24-hour alignment accuracies are greater with the clustered alignment method than with ordinary SCOW. (The exception is when $k = 1600$, which performs significantly worse.)

With queries containing fewer than three observations, the clustered alignment method actually results in somewhat less accurate alignments than the non-clustered method (i.e. ordinary SCOW). These results can be explained by a bias-variance tradeoff (Geman et al., 1992). The sparsity, noise, and variation of the data mean that it is difficult to compute accurate single-gene alignments. Aggregating genes into clusters has a regularization effect as this alignment error is averaged out (Bar-Joseph et al., 2003). The more genes there are in a cluster, the greater the regularization effect. Thus we want to find the ideal tradeoff between the high-bias approach of few clusters (or one cluster, in the limit), and the high-variance approach of many clusters. The variance component of the error is more significant in the case when the queries are short. We can conclude, however, that the clustered alignment approach demonstrates good predictive value for moderately sized queries and a range of values of k.

In our second experiment, we are interested in identifying sets of genes that are distorted in similar ways in a knockout experiment focusing on circadian rhythms. Mop3 is a transcription factor in hepatocytes (Bunger et al., 2000, 2005) that is a positive regulator of circadian rhythm and activates the transcription of genes such as Per1 and Tim. There are two sets of mice in this experiment. The control group has a functional Mop3 gene, whereas the knockout group does not. This is a time-course study based on *Zt* which stands for *Zeitgeiber time*—the number of hours after exposure to light begins. Before Zt0 the mice are kept in darkness for a period. At Zt0 the lights turn on, and at Zt12 they turn off again. At intervals of four hours from Zt0 to Zt20, three mice from each group are sacrificed, and microarrays are derived from pooled RNA samples from the livers of each set of mice. In all, 27,962 genes are measured. We interpolate the series with B-splines so that we can sample measurements every two hours. We use SCOW to align the time series, modified as described to perform double shorting. We concatenate the control group series with itself, and then allow the alignment to short at both ends with respect to that series but not at all with respect to the knockout series.

Figure 7.6 shows alignments for several genes in each cluster, as determined by our clustered alignment algorithm. Here we set the number of clusters $k = 5$. Each panel represents one of the clusters, and within each we show the three genes with the highest relative scores for that cluster.

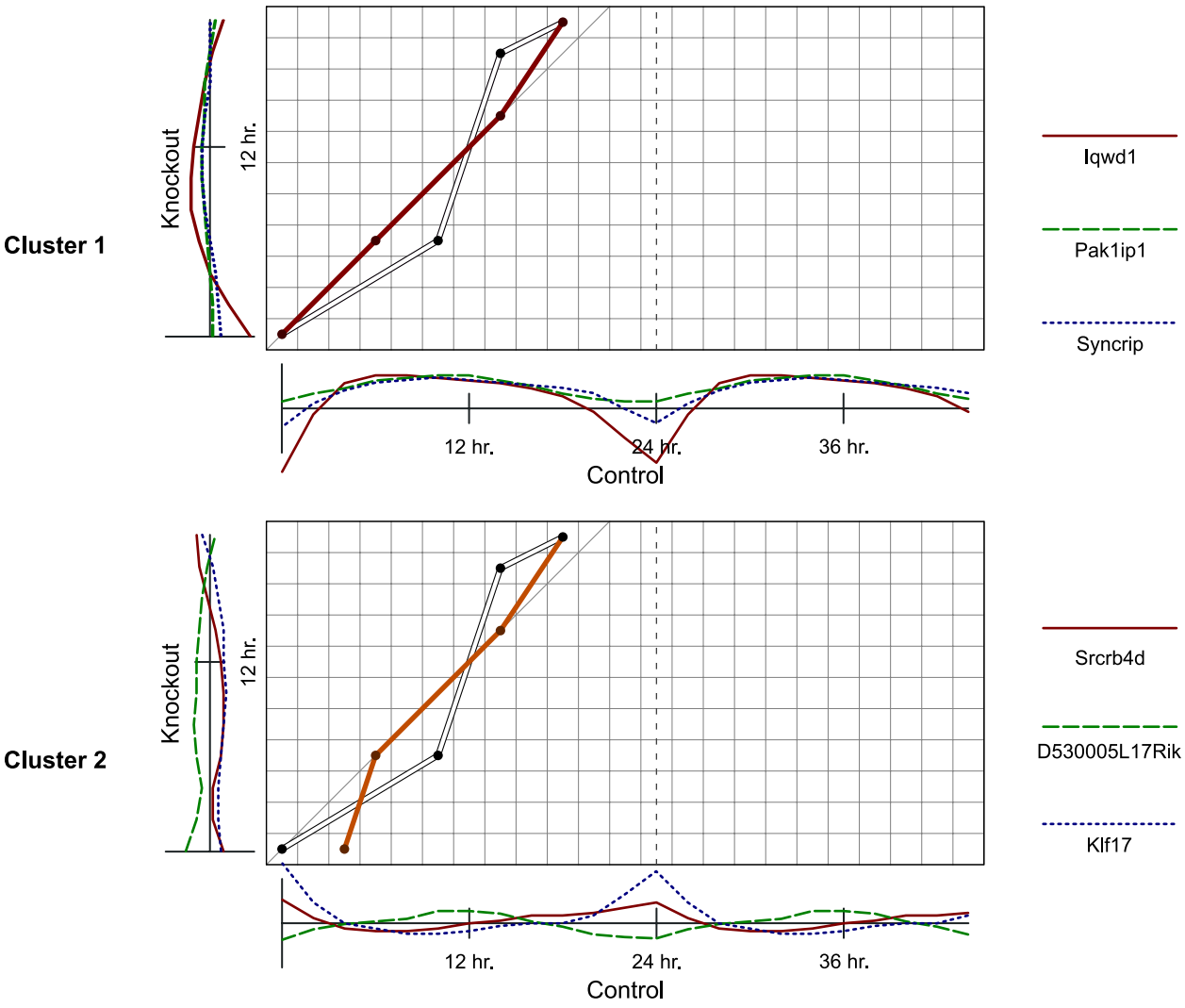

Figure 7.6: Alignment clusters found by our method for the Mop3-knockout circadian data. Each panel shows the top three genes for a different cluster. The white alignment paths represent the consensus alignment for all the genes, while the solid paths represent the cluster-specific alignments. (Figure continues on next page.)

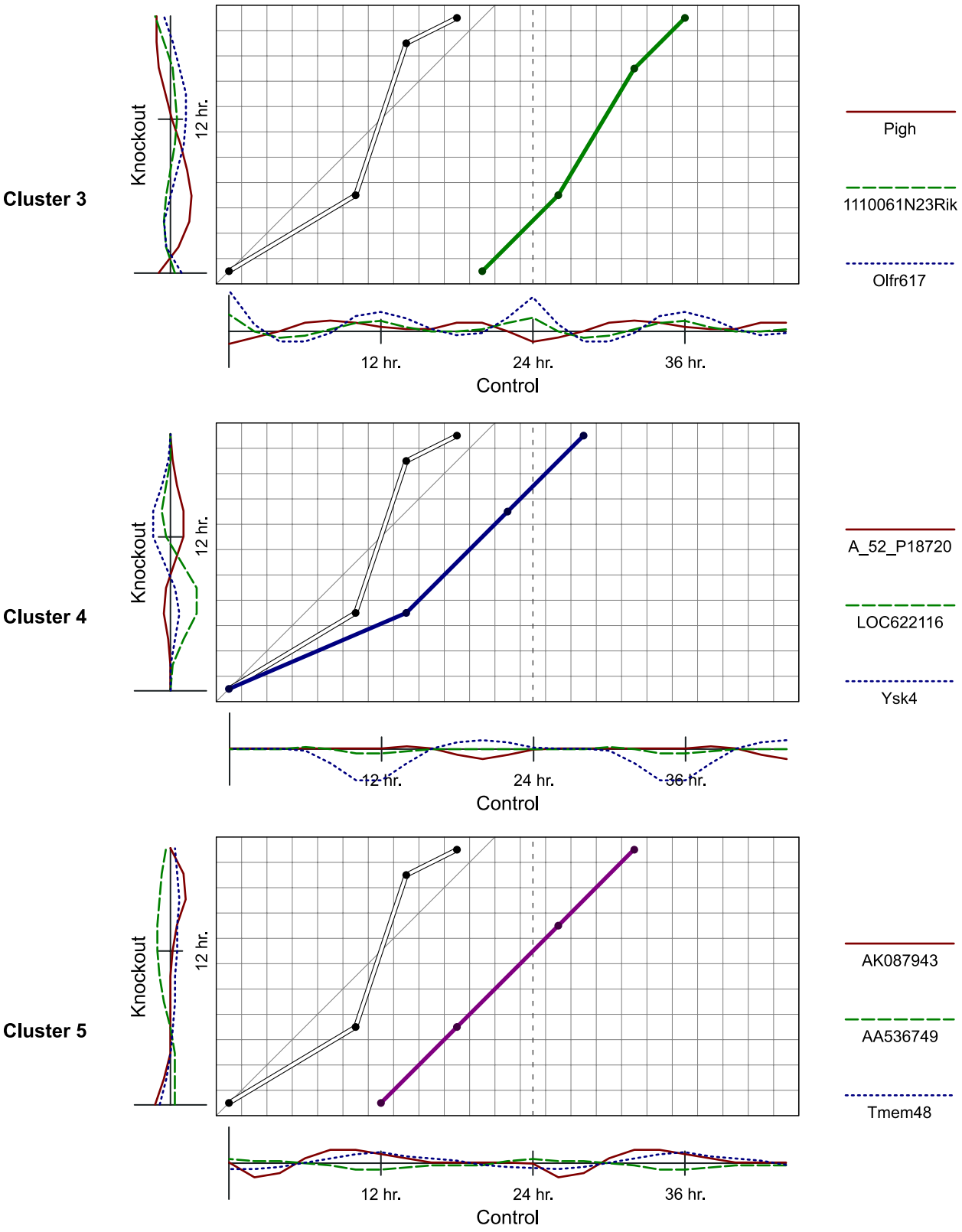

Figure 7.6 continued

The white alignment path in each plot represents the consensus alignment, when all genes are warped as a unit. The solid alignment path represents the cluster's individual alignment.

The clustered alignment allows us to uncover sets of genes that are disrupted in a similar manner by the knockout, even when their expression profiles are quite different. It is clear that the clustered alignments align the series better than the consensus alignment. Peaks and valleys in the expression data line up well for the solid cluster alignment paths, whereas they often do not for the white consensus ones. For example, the genes in the Cluster 5 have undergone a large phase shift. The consensus path often aligns segments with quite different expression profiles, whereas the cluster path shifts the starting point by 12 hours and achieves good agreement. In the Cluster 4 the genes appear to be acting more quickly in the knockout mice, while the consensus alignment would indicate they are acting more slowly. It should also be noted that often the genes within a cluster have very different expression profiles. Consider Cluster 4, in which the profiles for the three genes are all quite different, but the mapping between control and knockout is similar. This effect illustrates the advantage of clustering alignments in contrast to clustering the expression profiles directly.

7.4 Summary

In this chapter we have introduced a method for performing clustered alignments. This method represents a middle ground between aligning all genes together (which is an oversimplifying assumption), and aligning them all separately (which is prone to error). We use a technique similar to k-means clustering in order to partition genes into k distinct groups based on alignment. Previous methods (Aach and Church, 2001; Bar-Joseph et al., 2003; Criel and Tsiporkova, 2006) have focused on clustering by expression profile, but have not clustered based on alignment between two different treatments or conditions.

We have also shown a method to perform double-shorting with respect to a cyclic gene knockout experiment. Simply shorting both ends can create very small alignment paths that can overestimate the similarity between very different series. Our method addresses this issue by doubling

the series most likely to be periodic, and then allowing the alignment to short on both ends only with respect to this series.

We have also performed experiments that show the utility of our clustered alignment method. We have shown that when different groups of genes are aligned differently, our method matches and aligns queries more accurately than either warping all genes separately or warping them all together. In addition, we have applied our clustered alignment method to time series from a circadian knockout experiment, and have shown that our clustered alignments result in more accurate alignments for several genes.

Chapter 8

Conclusions

We conclude by summarizing our contributions, and discussing some possible areas of future work.

8.1 Summary of Contributions

There are several contributions this dissertation has made to the state of the art in alignment of gene-expression time-series data. We have shown that our techniques improve both alignment and similarity searches when using such data. This helps toward our high-level goals of creating an online tool for comparing time series, and elucidating the effects of particular treatments on the expression levels of individual genes.

- *Smoothing splines:* Previous methods, which have used B-splines to reconstruct unosbserved data (Bar-Joseph et al., 2003; Luan and Li, 2004), fit high-order splines in such a way that they may vary radically in order to exactly intercept every observed data point. We have relaxed this assumption by using the coefficients (control points) from lower-order splines. This creates splines that tends to stay within the convex hull of the observed points, and are much more resistant to outlying data points. These properties are important, given that gene-expression time-series data is temporally sparse, and subject to both technical noise and biological variability. Our experiments in Chapter 4 showed that reconstructing series with smoothing splines results in higher alignment and treatment accuracy than when using traditional intercepting splines.

- *Observation points as spline knots:* The same studies that have used splines have also assumed that the knots, or points of discontinuity, of the splines should be evenly distributed in time. This can result in unsolvable linear equations if the observed points are too far away from each other. Our solution is to use the observed times themselves as the knots. This guarantees that the equations can be solved, and focuses discontinuity in the interpolating splines on precisely the times that the biologists who gathered the data thought were interesting enough to measure.

- *Application of segment-based alignment methods to gene-expression data:* Previously, segment-based alignment methods were confined to the domain of chromatography. They are especially well suited to gene-expression time-series data for two reasons. First, distortions of adjacent times are not assumed to be independent of each other, so long as they are in the same segment. Second, the segment-based nature of the alignment path restricts the alignment, keeping it from being overfit to sparse data, while allowing more flexibility than parametric alignment approaches.

- *Shorting:* We have devised warping methods that can short an alignment, leaving the end of one series or the other unaligned. Shorting is a special case of local alignment, and is important in domains in which the ends of the time series should not necessarily be aligned. Our experiments in Chapter 5 showed that alignment methods that can short have an advantage over those that cannot.

- *Double-shorting:* We have developed an algorithm to allow double-shorting which entails leaving the beginning of one of the series unaligned, and the end of one of the series unaligned. This is important in domains wherein we cannot be sure that the first points of the time series are aligned.

- *Multisegment generative alignment:* This alignment method is based on a generative scoring function. It uses dynamic programming to execute a complete search of possible segments, returning the highest-scoring contiguous set. The method has higher treatment and alignment

accuracies than previous methods (e.g. dynamic time warping, parametric time warping) when performing similarity searches on the EDGE toxicology data. However it is slow, with a time complexity of $O(n^5)$ where n is the length of one of the time series.

- *Shorting correlation-optimized warping*: SCOW is another segment-based alignment method that we developed to counter the shortcomings of COW, a previous alignment algorithm. SCOW's scoring function is based on the Pearson cross correlation. SCOW does not perform a complete search, but uses a heuristic wherein it alternates searching for segment boundaries with respect to one series at a time. Its time complexity is $O(imn^3)$, where i is the number of one-dimensional iterations done, m is the number of segments, and n is the length of the time series. Its accuracy is slightly higher than the multisegment generative method, with respect to similarity searches on the EDGE data.

- *Experimental methodology for evaluating alignment methods:* We resample the time series so that both query and database series observations are regularly spaced. We also developed a technique to better estimate the treatment accuracy of a similarity search algorithm. Because in practice there will not be an exact match to the query within the database, we apply temporal distortions to the query series so that it does not precisely match a series in the database.

- *Heuristics for the multisegment generative method:* Because of its slow running time, we devised several heuristics to speed up the calculations in the multisegment generative method. The *cone filter heuristic* works by restricting the search space to a cone around the diagonal. It speeds up the algorithm by a constant amount, and in some cases helps accuracy by disallowing radical alignments. The *hybrid DTW filter heuristic* performs a first pass with a method similar to DTW, and then restricts the multisegment alignment to points close to the path so found. It speeds the method to $O(n^3)$. The *alternating search heuristic* exhibits the best speedup and the least effect on accuracy. It works by using the SCOW search technique to find segments, and like SCOW works in $O(imn^3)$ time.

- *Clustered alignments:* We have developed a method, based on k-means clustering, to simultaneously align and cluster genes. Our clusters are based on alignments between two treatments or conditions, rather than being based directly on the expression levels of the genes as in numerous previous studies (Chudova et al., 2003; Ernst et al., 2005; Gaffney and Smyth, 2005). This can be more useful in elucidating the genes' dependencies on one another, because closely related genes required by similar processes may have very different expression profiles. We also showed that alignments using this method can be more accurate when the genes vary between treatments in different ways.

8.2 Future Directions and Unsolved Problems

There are several outstanding problems that we believe should be addressed in future research.

- *Automatic determination of the number of clusters:* With our current clustered alignment method, the user must determine the number of clusters k a priori. This will be problematic if the user has no prior idea of how many clusters to expect. Background knowledge might begin to play a role in this, as more of the gene dependency networks in model organisms are mapped out. Alternatively, we could use model selection strategies such as leave-one-out with cross validation to estimate what k should be.

- *Automatic determination of spline order:* Likewise, the order of the B-splines used for reconstruction is a parameter set by the user. It would be preferable to have a way to determine this automatically. As with the number of clusters, a leave-one-out method might be useful for determining spline order. Such a method might also incorporate domain knowledge for reconstruction, as do Nachman et al. (2004), Farina et al. (2008), and Chechik and Koller (2009).

- *Mixture model for clustering:* Our clustering method uses hard clustering, in which every gene has membership in one and only one cluster. It might be beneficial to use soft clustering, in which every gene is a member of multiple clusters with different probabilities. However, our initial attempts to do this were not successful. The algorithm tended to give most genes

an equal chance of belonging to all clusters. In the analogous problem of clustering points in $\mathbb{R}^n$ using EM (Dempster et al., 1977), this often happens when the sizes of clusters (i.e. the deviations from the clusters' centers) are not allowed to vary. If we were to develop an analog of deviation with respect to our alignment clusters, this might help the quality of our alignments.

- *Further development of double-shorting:* We took advantage of idiosyncracies in the Mop3 knockout problem when we implemented double-shorting. With the gene in one treatment being cyclic, we only need to short with respect to that treatment. However, without a cyclic gene, we would need to allow shorting in both dimensions. It is unclear that our methodology would find a good alignment path in such a case. The risk is that double-shorting would allow very short alignment paths, and the shorter the path, the more likely the series are to be alike there by coincidence. It might help to bias the search toward longer alignment paths, or to find a well-aligned "seed" point in alignment space from which to search in both directions.

- *Explanation of differences between SCOW and multisegment generative methods:* Our experiments show that SCOW's treatment and alignment accuracies are higher than those of the multisegment generative method when choosing among several treatments, but that the multisegment generative's alignment accuracy can be higher when we align without having to pick a treatment. Explaining this dichotomy could give us a better idea of when to use which method.

- *Alignments in which expression is a function of space:* We have restricted ourselves to cases in which the expression of each gene is a function of only time. However, in some cases, it might be a function of space as well (e.g. in a two-dimensional cell culture in which a treatment is applied to a single point and allowed to spread). Extending our warping algorithms to align in both time and space could help compare experiment results in these domains.

8.3 Final Words

In this dissertation, we have explored the alignment of gene-expression time series using a variety of methods, including two of our own creation: SCOW and the multisegment generative method. We have helped to answer several related questions, including:

- *How should we reconstruct temporally-sparse gene-expression time-series data?* Smoothing B-splines provide an accurate method for interpolation.
- *Should genes be aligned globally, or should methods that allow non-global alignments (e.g. shorted alignments) be preferred?* Alignment methods that short have a natural advantage over those that do not.
- *Should genes all be aligned separately, as a unit, or in independent clusters?* Aligning them in clusters allows different genes to have different alignments, while still averaging out the alignment error that results from temporal sparsity.
- *What is an appropriate space of alignments to consider to achieve accurate alignments of sparse gene-expression time series?* A segment-based alignment is expressive enough to represent most distortions seen, while not being too expressive for temporally sparse gene-expression data.

It is our hope that these findings which we have presented will help others answer future questions in the alignment and similarity of gene-expression time-series data.

Bibliography

J. Aach and G. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

S. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219:555–565, 1991.

S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

V. Aris, M. Cody, J. Cheng, J. Dermody, P. Soteropoulos, M. Recce, and P. Tolias. Noise filtering and nonparametric analysis of microarray data underscores discriminating markers of oral, prostate, lung, ovarian and breast cancer. *BMC Bioinformatics*, 5(1):185, 2004.

Z. Bar-Joseph, G. Gerber, D. Gifford, T. Jaakkola, and I. Simon. Continuous representations of time-series expression data. *Journal of Computational Biology*, 10(3-4):341–356, 2003.

M. Barenco, J. Stark, D. Brewer, D. Tomescu, R. Callard, and M. Hubank. Correction of scaling mismatches in oligonucleotide microarray data. *BMC Bioinformatics*, 7(1):251, 2006.

A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

B. Bollobás, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated geometric sets. *Nordic Journal of Computing*, 8(4):409–423, 2001.

M.K. Bunger, J.A. Walisser, R. Sullivan, P.A. Manley, S.M. Moran, V.L. Kalscheur, R.J. Colman, and C.A. Bradfield. Progressive arthropathy in mice with a targeted disruption of the Mop3/Bmal-1 locus. *Genesis*, 41(3):122–32, 2005.

M.K. Bunger, L.D. Wilsbacher, S.M. Moran, C. Clendenin, L.A. Radcliffe, J.B. Hogenesch, M.C. Simon, J.S. Takahashi, and C.A. Bradfield. Mop3 Is an Essential Component of the Master Circadian Pacemaker in Mammals. *Cell*, 103(7):1009–1017, 2000.

E.G. Caiani, A. Porta, G. Baselli, M. Turiel, S. Muzzupappa, F. Pieruzzi, C. Crema, A. Malliani, S. Cerutti, and D. di Bioingegneria. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. *Computers in Cardiology 1998*, pages 73–76, 1998.

H.Y. Chang, J.A. Thomson, and X. Chen. Microarray analysis of stem cells and differentiation. *Methods in enzymology*, 420:225–254, 2006.

G. Chechik and D. Koller. Timing of gene expression responses to environmental changes. *Journal of Computational Biology*, 16(2):279–290, 2009.

D. Chen, P. Wang, R.L. Lewis, C.A. Daigh, C. Ho, X. Chen, J.A. Thomson, and C. Kendziorski. A microarray analysis of the emergence of embryonic definitive hematopoiesis. *Experimental Hematology*, 35(9):1344–1357, 2007.

D. Chudova, S.J. Gaffney, and P. Smyth. Probabilistic models for joint clustering and timewarping of multi-dimensional curves. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, volume 10, pages 134–141. Morgan Kaufmann, 2003.

A. Claridge-Chang, H. Wijnen, F. Naef, C. Boothroyd, N. Rajewsky, and M.W. Young. Circadian regulation of gene expression systems in the Drosophila head. *Neuron*, 32(4):657–672, 2001.

F. Coolidge. *Statistics: A Gentle Introduction, Second Edition*. Sage Publications, 2006.

J. Criel and E. Tsiporkova. Gene time expression warper: A tool for alignment, template matching and visualization of gene expression time series. *Bioinformatics*, 22(2):251–252, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.

P.H.C. Eilers. Parametric time warping. *Analytical Chemistry*, 76(2):404–411, 2004.

J. Ernst, G. Nau, and Z. Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(Suppl. 1):i159–i68, 2005.

V. Eruhimov, V. Martyanov, and E. Tuv. Constructing high dimensional feature space for time series classification. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pages 414–421. Springer-Verlag Berlin, Heidelberg, 2007.

W. Euachongprasit and C.A. Ratanamahatana. Accurate and efficient retrieval of multimedia time series data under uniform scaling and time warping. In *Advances in Knowledge Discovery and Data Mining: 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20-23, 2008 Proceedings*, pages 100–111. Springer, 2008.

L. Farina, A. De Santis, S. Salvucci, G. Morelli, and I. Ruberti. Embedding mRNA stability in correlation analysis of time-series gene expression data. *PLoS Computational Biology*, 4(8), 2008.

A.W. Fu, E. Keogh, L.Y.H. Lau, C.A. Ratanamahatana, and R.C.W. Wong. Scaling and time warping in time series querying. *The International Journal on Very Large Data Bases*, 17(4): 899–921, 2008.

S.J. Gaffney and P. Smyth. Joint probabilistic curve clustering and alignment. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 473–480. MIT Press, 2005.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

K. Gollmer and C. Posten. Detection of distorted pattern using dynamic time warping algorithm and application for supervision of bioprocesses. *On-Line Fault Detection and Supervision in Chemical Process Industries*, 1995.

K. Hayes, A. Vollrath, G. Zastrow, B. McMillan, M. Craven, S. Jovanovich, J. Walisser, D. Rank, S. Penn, J. Reddy, R. Thomas, and C. Bradfield. EDGE: A centralized resource for the comparison, analysis and distribution of toxicogenomic information. *Molecular Pharmacology*, 67(4): 1360–1368, 2005.

F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

M.W. Kadous. Learning comprehensible descriptions of multivariate time series. In *Proceedings of the 16th International Conference on Machine Learning*, pages 454–463. Morgan Kaufmann, 1999.

E.J. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 406–417. Morgan Kaufmann, 2002.

E.J. Keogh. Efficiently finding arbitrarily scaled patterns in massive time series databases. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 253–265. Springer, 2003.

E.J. Keogh and M.J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289. ACM New York, NY, USA, 2000.

A.B. Khodursky, B.J. Peter, N.R. Cozzarelli, D. Botstein, P.O. Brown, and C. Yanofsky. DNA microarray analysis of gene expression in response to physiological and genetic changes that affect tryptophan metabolism in Escherichia coli. *PNAS*, 97(22), 2000.

J. Lamb, E. Crawford, D. Peck, J. Modell, I. Blat, M. Wrobel, J. Lerner, J.-P. Brunet, A. Subramanian, K. Ross, M. Reich, H. Hieronymus, G. Wei, S. Armstrong, S. Haggerty, P. Clemons, R. Wei, S. Carr, E. Lander, and T. Golub. The connectivity map: Using gene-expression signatures to connect small molecules, genes and disease. *Science*, 313:1929–1935, 2006.

C.J. Langmead, C.R. McClung, and B.R. Donald. A maximum entropy algorithm for rhythmic analysis of genome-wide expression patterns. In *Proceedings of the 2002 IEEE Computer Society Bioinformatics Conference*, volume 1, pages 237–245. IEEE Computer Society, 2002a.

C.J. Langmead, A.K. Yan, C.R. McClung, and B. R. Donald. Phase-independent rhythmic analysis of genome-wide expression patterns. In *Proceedings of the 6th Annual International Conference on Computational Biology (RECOMB)*, pages 205–215. ACM, 2002b.

X. Leng and H.-G. Müller. Classification using functional data analysis for temporal gene expression data. *Bioinformatics*, 22(1):68–76, 2006.

J. Listgarten, R. Neal, S. Roweis, and A. Emili. Multiple alignment of continuous time series. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 817–824. MIT Press, 2005.

X. Liu and H.-G. Müller. Modes and clustering for time-warped gene expression profile data. *Bioinformatics*, 19(15):1937–1944, 2003.

Y. Luan and H. Li. Model-based methods for identifying periodically expressed genes based on time course microarray gene expression data. *Bioinformatics*, 20(3):332–339, 2004.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

M.D. Morse and J.M. Patel. An efficient and accurate method for evaluating time series similarity. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 569–580. ACM, 2007.

I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(Suppl. 1):i248–i256, 2004.

G Natsoulis, L. El Ghaoui, G. Lanckriet, A. Tolley, F. Leroy, S. Dunlea, B. Eynon, C. Pearson, S. Tugendreich, and K. Jarnagin. Classification of a large microarray data set: Algorithm comparison and analysis of drug signatures. *Genome Research*, 15(5):724–736, 2005.

N. V. Nielsen, J. M. Carstensen, and J. Smedsgaard. Aligning of single and multiple wavelength chromatographic profiles for chemometric data analysis using correlation optimised warping. *Journal of Chromatography A*, 805:17–35, 1998.

I. Ong, J. Glasner, and D. Page. Modelling regulatory pathways in E. coli from time series expression profiles. *Bioinformatics*, 18(Suppl. 1):S241–S248, 2002.

C. Ratanamahatana and E.J. Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining*, pages 506–510. SIAM, 2005.

D. Rogers and J. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, 1989.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE ASSP Magazine*, 26:43–49, 1978.

M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

M.D. Schmill, T. Oates, and P.R. Cohen. Learned models for continuous planning. In *In Proceedings of Uncertainty 99: The 7th International Workshop on Artificial Intelligence and Statistics*, pages 278–282. Morgan Kaufmann, 1999.

A.A. Smith and M. Craven. Fast multisegment alignments for temporal expression profiles. In *Proceedings of the 7th International Conference on Computational Systems Bioinformatics*, volume 7, pages 315–326. Imperial College Press, 2008.

A.A. Smith, A. Vollrath, C.A. Bradfield, and M. Craven. Similarity queries for temporal toxicogenomic expression profiles. *PLoS Computational Biology*, 4(7):e1000116, Jul 2008.

A.A. Smith, A. Vollrath, C.A. Bradfield, and M. Craven. Clustered alignments of gene-expression time series data. *Bioinformatics*, 25, 2009.

P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular biology of the cell*, 9(12): 3273–3297, 1998.

K.F. Storch, O. Lipan, I. Leykin, N. Viswanathan, F.C. Davis, W.H. Wong, and C.J. Weitz. Extensive and divergent circadian gene expression in liver and heart. *Nature*, 417(6884):78–83, 2002.

R. Thomas, D. Rank, S. Penn, G. Zastrow, K. Hay, K. Pande, E. Glover, T. Silander, M. Craven, J. Reddy, S. Jovanovich, and C. Bradfield. Identification of toxicologically predictive gene sets using cDNA microarrays. *Molecular Pharmacology*, 60(6):1189–1194, 2001.

J. Tuke, G.F.V. Glonek, and P.J. Solomon. Gene profiling for determining pluripotent genes in a time course microarray experiment. *Biostatistics*, 10(1):80, 2009.

V.E. Velculescu, L. Zhang, B. Vogelstein, and K.W. Kinzler. Serial analysis of gene expression. *Science*, 270(5235):484–487, 1995.

Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 2008.

R.F. Weaver. *Molecular Biology, Second Edition*. McGraw-Hill, 2002.

X. Xi, E. Keogh, C. Shelton, L. Wei, and C.A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd International Conference on Machine learning*, pages 1033–1040. ACM, 2006.

Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi. Decision-tree induction from time-series data based on a standard-example split test. In *Proceedings of the 20th International Conference on Machine Learning*, volume 20, pages 840–847, 2003.

M. Yuan and C. Kendziorski. Hidden Markov models for microarray time course data in multiple biological conditions. *Journal of the American Statistical Association*, 101(476):1323–1332, 2006a.

M. Yuan and C. Kendziorski. A unified approach for simultaneous gene clustering and differential expression identification. *Biometrics*, 62(4):1089–1098, 2006b.

APPENDIX
Implementation Notes

Algorithms were implemented in Java, with supplementary code in Python. Code can be downloaded from `http://www.biostat.wisc.edu/~aasmith/catcode`.

This dissertation was written using LaTeX, on Linux-based computers. Figures were designed with Inkscape.