A Novel Computational Method for Inferring Dynamic Genetic Regulatory Trajectories

by

Christopher Campbell Reeder

B.A., Computer Science, UC Berkeley (2006)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author . . . .
Department of Electrical Engineering and Computer Science
August 8, 2008

Certified by . . . .
David K. Gifford
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

A Novel Computational Method for Inferring Dynamic Genetic Regulatory Trajectories

by

Christopher Campbell Reeder

Submitted to the Department of Electrical Engineering and Computer Science
on August 8, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

We present a novel method called Time Series Affinity Propagation (TSAP) for inferring regulatory states and trajectories from time series genomic data. This method builds on the Affinity Propagation method of Frey and Dueck [10]. TSAP incorporates temporal constraints to more accurately model the dynamic nature of underlying biological mechanisms. We first apply TSAP to synthetic data and demonstrate its ability to recover underlying structure that is obscured by noise. We then apply TSAP to real data and demonstrate that it provides insight into the relationship between gene expression and histone post-translational modifications during motor neuron development. In particular, the trajectories taken by the Hox genes through the space of regulatory states are characterized. Understanding the dynamics of Hox regulation is important because the Hox genes play a fundamental role in the establishment of motor neuron sub-type identity during development [6].

Thesis Supervisor: David K. Gifford
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to first thank Dr. Georg Gerber. Upon my arrival at MIT, he immediately treated me as a collaborator. I am very grateful for his willingness to discuss research ideas for hours on end. I owe much of my growth as a researcher to my interactions with Georg. Indeed, most of the work that led to this thesis was based on ideas developed during our conversations.

I would also like to acknowledge my research advisor Professor David Gifford. He has an uncanny ability to identify important research problems. The manner in which he has led the effort to bridge the gap between biology and computer science is truly inspiring.

I am very grateful to those who have been in the Gifford Lab during my time here so far: Bob Altshuler, Tim Danford, Dr. Robin Dowell, Yuchun Guo, Dr. Shaun Mahony, Alex Rolfe, and Alex Tsankov. It is wonderful to be surrounded by such intelligent people. Shaun was particularly helpful in helping me figure out how to interpret and visualize the results of my thesis work. Also, Tim and Alex Rolfe have built an infrastructure for working with biological data that has been an indispensable resource. I am also grateful to Jeanne Darling, without whom the workings of the Gifford Lab would surely grind to a halt.

My friends here at MIT, from Berkeley, and from back home have been a great source of support and distraction, both of which have been much needed at times. Finally, I must thank my parents for their love and for giving me such a great gift in that they have always encouraged me in my various aspirations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description

Adult multicellular organisms can consist of hundreds of distinct cell types. Although all of the cells in an individual organism, with very few exceptions, contain the same set of genes, cells of different cell types establish and maintain their identity by expressing different subsets of genes [11]. Cells are unable to function properly unless the correct set of cell type specific genes are expressed. Thus, the precise control of gene expression is crucial to the ability of multicellular organisms to live.

Biology seeks to understand the manner in which cell type specific expression patterns are established during the development of a multicellular organism. The origin of all cells in a multicellular organism can be traced back to a single cell, the fertilized egg. Through differentiation, cells give rise to progeny of more specialized cell types. This process begins with the fertilized egg and continues through several intermediate stages, ultimately producing all of the cells in the adult organism. External signals trigger changes in the regulatory mechanisms that control gene expression to induce differentiation. The complexity of the regulatory mechanisms that control gene expression and the fact the the genomes of many multicellular organisms contain tens of thousands of genes makes understanding differentiation at the level of gene regulation a difficult problem.

In recent years, various technologies have been developed that allow many aspects of gene regulation to be measured on a genome-wide scale. For example, DNA microarrays were originally developed to measure levels of RNA transcripts, an intermediate product of gene expression, corresponding to thousands of genes simultaneously [23]. Such technologies provide an opportunity to characterize the regulatory mechanisms that are at work in different cell types. However, the data sets produced by genome-wide technologies are large and contain significant experimental noise, thus making them difficult to interpret. Because of this difficulty, the development and application of computational methods has become a vital part of research in the field of gene regulation.

We present a novel computational method called Time Series Affinity Propagation (TSAP) to investigate the regulatory mechanisms involved in the establishment of cell type specific expression patterns. Many previous computational approaches to the analysis of genomic data have not addressed the availability of multiple, heterogeneous sources of data, nor have they taken into account the inherent temporal structure of data related to development. TSAP has been designed to address these issues. The derivation of TSAP will be presented as well as novel results obtained through the use of TSAP to study motor neuron development.

1.2 Biological Background

TSAP is designed to be able to incorporate measurements of any sort of genomic phenomena. However, the discussion in this thesis will focus on genomic data of two types. The first type is expression data collected using DNA microarray technology. The second type is genome-wide location mapping data for histone post-translational modifications (PTMs) collected using chromatin immunoprecipitation combined with microarray technology [21].

1.2.1 Gene Expression

Contained within the cells of all known living organisms is a substance called deoxyribonucleic acid (DNA). The basic unit of DNA is the nucleotide, which consists of a nitrogenous base, a sugar, and a phosphate group. DNA is most commonly found to exist in cells as polymers of nucleotides with the phosphate residue of one nucleotide binding to the sugar residue of the next nucleotide. The nitrogenous bases found in living cells are generally one of four structures: Adenine (A), Thymine (T), Guanine (G), or Cytosine (C). The ordering of these bases along the nucleotide strand encodes the information for producing proteins as well as other information necessary for the proper functioning of the cell.

A portion of DNA sequence that encodes information for producing a protein (or possibly a set of related proteins in the case of alternative splicing) is referred to as a protein-coding gene. A protein-coding gene is "expressed" when copies of the protein coded for by the gene are produced in the cell. Expression of a protein-coding gene involves the creation of an intermediate substrate. Molecules of a substance closely related to DNA called RNA are "transcribed" from the DNA sequence of the protein-coding gene. The RNA molecule consists of a sequence of nucleotides that are complementary to the nucleotides of the DNA sequence of the gene. The information carried by these RNA molecules is then "translated" by ribosomal complexes into protein.

Several technologies, such as DNA microarrays, measure RNA transcript levels rather than measure protein levels directly. DNA microarray technology takes advantage of the fact that single stranded nucleic acid molecules that have complementary sequences will tend to hybridize to one another. Microarrays contain many single stranded nucleic acid molecules called probes that are attached to a glass slide. The probes are designed to contain sequences corresponding to locations of interest in the genome. For the purpose of measuring gene expression, the probes consist of parts of the sequences of genes. RNA molecules are extracted from a sample of cells that are of the cell type of interest. The RNA molecules are amplified and labeled and then allowed to interact with the probes on the microarray. The technology relies on the tendency of nucleic acid fragments to hybridize to the most perfectly complementary probes which should correspond to the genes from which the RNA molecules were transcribed. However, nucleic acid fragments will sometimes hybridize to probes that are not perfectly complementary. Also, the efficiency of hybridization depends on the actual bases that make up the probe sequences. An optical scanner is used to detect where labeled molecules have hybridized to probes on the microarray and in what concentration. Image analysis is applied to the image obtained from the scanner to obtain a measurement of how many RNA molecules corresponding to each gene were present in the sample of cells.

1.2.2 Histone Post-Translational Modifications

The DNA in a cell exists in a complex called chromatin. This complex contains proteins, most prominently histone proteins, in addition to DNA. The histone proteins and DNA form structures called nucleosomes in which 147 base pairs-worth of DNA wraps around a histone octamer [22]. The histone proteins are often subject to post-translational modifications (PTMs), usually on their N-terminal tails which extend out from the nucleosome complex. It is clear that there is some relationship between histone PTMs and gene regulation and this relationship has been the focus of much recent research [17].

To measure the locations in the genome where histones with particular PTMs exist, a combination of DNA microarray technology (chip) preceded by chromatin immunoprecipitation (ChIP) may be utilized [14]. Collectively, this approach will be referred to as ChIP-chip. The general idea is to isolate fragments of DNA that are associated with a histone PTM of interest and then infer the location of these fragments in the genome according to the probes to which they hybridize on a DNA microarray. To perform ChIP, live cells are treated with a chemical, often formaldehyde, which causes proteins that are interacting with DNA to become cross-linked to the DNA. The resulting DNA-protein complex can then be extracted from the cells intact. The DNA is fragmented using a method such as sonication. The next step requires the availability of an antibody protein that binds specifically to histones with the PTM of interest. Through immunoprecipitation, fragments of DNA cross-linked to histones with the PTM of interest are isolated out of the fragmented DNA-protein complex that was extracted from the cells. The DNA fragments can then be purified from the precipitated mixture of DNA, histones, and antibodies. The locations of these fragments in the genome can be measured using a DNA microarray experiment in essentially the same manner that the location of genes associated with RNA transcripts are measured in a gene expression experiment. It is often useful to use a special type of DNA microarray called a tiling array in which probes are tiled across the genome at a regular interval. Thus, the location of the modified histone can be measure with greater spatial resolution than if microarrays were used that contain only a handful of probes per gene.

Two histone PTMs in particular will be discussed in this thesis. Histone H3 lysine 27 trimethylation (H3K27me3) and histone H3 lysine 4 trimethylation (H3K4me3) are two PTMs that involve the methylation of different lysine residues on the N-terminal tail of histone 3. Most evidence to date has associated H3K27me3 with repression of gene expression and H3K4me3 with activation of gene expression. TSAP is applied in this thesis to provide a unique view of the temporal relationship between these histone PTMs and gene expression during development.

1.3 Solution Overview

The objective of TSAP is twofold. The first part of TSAP's objective is to discover gene regulatory states that play a prominent role in the establishment of cell type specific expression patterns. For the purposes of this thesis, a gene regulatory state is defined as the joint occurrence of a specific gene expression level and associated histone PTM patterns at a given gene. The second part of TSAP's objective is to trace the trajectories of genes as they move between regulatory states during the establishment of cell type specific expression patterns.

The approach taken by TSAP to the first part of the objective is to find a partitioning of the genes under consideration such that each partition contains genes that exist in a common regulatory state. An assumption is made that genes that exist in a common regulatory state will exhibit similar levels of gene expression and will be associated with histones that exhibit similar patterns of modification. Thus, TSAP finds a partitioning of the genes under consideration that maximizes the similarity within each partition based on gene expression and histone modification.

To achieve the second part of the objective, TSAP allows each gene to be assigned to different partitions as cells pass through different cell types during differentiation. However, since the regulatory state of a gene is not expected to be independent in related cell types, the partitioning of genes for a particular cell type is influenced by the partitioning of genes for related cell types.

1.4 Related Work

TSAP is related to the Affinity Propagation (AP) clustering method [10]. AP approaches the well established K-medians clustering problem from a statistical inference perspective. A probability distribution is defined for which finding a maximal configuration is equivalent to finding an optimal solution to the K-medians problem. A variant of belief propagation [20] is used to find an approximately maximal configuration for the probability distribution. A more detailed discussion of AP and TSAP is presented in Chapter 2. No efficient method for finding an exact solution to the K-medians problem is known. In fact, the problem is known to be NP-hard [5]. Several approximation algorithms for the K-medians problem have been developed that are based on solving a linear programming (LP) relaxation. The theoretical properties of belief propagation are not well understood, so the LP based approaches have an advantage to AP in that they come with well defined theoretical guarantees. However, AP is very practical to implement and has been shown to converge efficiently to good solutions in practice. Further justification for the use of a method related to AP is given in Chapter 2.

The K-means problem is another formulation of the clustering problem that is related to the K-medians problem. In the K-medians problem, the objective is to partition the data into K clusters such that each cluster is centered around a “median” point from the data set. In the K-means problem, the objective is the same except the point that each cluster is centered around need not be a point from the data set. One approach to finding an approximate solution to the K-means problem is the iterative algorithm that is commonly referred to as the K-means algorithm. Another related approach to clustering is the use of mixture models. In this approach, each cluster is associated with a probability distribution rather than just a particular point. It can be shown that the K-means algorithm is a nonprobabilistic limit of the Expectation Maximization algorithm applied to mixtures of Gaussian distributions [3].

Another approach to clustering that is very common, especially among biological applications, is hierarchical clustering. Rather than partition the data into some K number of clusters, hierarchical clustering methods find a tree structure that represents the similarity of points in the data set. The root of such a tree represents the data set as a whole. At each branch of the tree a set of points is split into smaller sets in a way that reflects a greater degree of similarity between the points within the subsets than between points in different subsets. The leaves of the tree represent the individual points from the data set. In one of the first applications of a clustering method to genomic data, a hierarchical clustering method was used to cluster gene expression data measured by DNA microarrays using material from *Saccharomyces cerevisiae* [8]. Examples of other clustering methods that have been applied to genomic data include Self Organizing Map and Principal Components Analysis. Handl et al. [13] provide a more thorough survey of recent applications of clustering methods to genomic data.

The clustering methods mentioned thus far, other than TSAP, do not explicitly take into account any prior knowledge about relationships between data points. Some methods have been developed for genomic applications that do explicitly take into account the dynamic nature of many genomic data sets. See [9] for a more thorough discussion of previous approaches to analyzing genomic data that have temporal structure. One approach in particular that is similar in a sense to TSAP is the use of Hidden Markov Models (HMMs) to cluster data [16]. In this approach, K states are posited to exist in the data and the trajectories taken by data points through these states are traced. TSAP has several advantages over such an approach including having the value of K determined in part by the data themselves and the fact that the states found by TSAP need not fit an underlying parametric distribution. A variation on HMMs called Infinite Hidden Markov Models [1] avoid the requirement of having to choose K ahead of time. However, states are still assumed to have been generated by parametric distributions.

1.5 Thesis Overview

In Chapter 2 of this thesis, we discuss the use of clustering methods for discovering regulatory states and trajectories. We will describe the Affinity Propagation (AP) algorithm, as well as briefly introduce graphical models and belief propagation. This will lead to a derivation of Time Series Affinity Propagation which is an extension of AP that explicitly models the temporal structure of dynamic data. We will suggest an approach to constructing a similarity measure from dynamic gene expression and histone PTM data. Chapter 2 concludes with a description of an approach to visualizing the states and trajectories that are found by TSAP.

In Chapter 3 we present applications of TSAP to three data sets. The first two data sets consist of synthetic data. We will demonstrate with the first synthetic data set TSAP's performance in a simplistic setting. We constructed the second synthetic data set to simulate a real biological data set. TSAP's performance on this synthetic data set motivates an application to a real data set. Thus, we apply TSAP to data from a particular mouse developmental system. We discuss the relationship between the histone PTMs and gene expression in the context of this biological system.

In the final chapter, we make some concluding remarks and present ideas for future related work.

Chapter 2

State Inference

2.1 Clustering

Given data for a set of genes, the objective of gene regulatory state inference is to discover a set of states that classify genes in a biologically relevant manner. One approach to achieving this objective is to organize the set of genes into groups such that genes within groups tend to be more similar than genes from different groups according to a notion of similarity based on the data. Methods that take this approach to arbitrary data sets have been discussed extensively in the machine learning and statistics literature. Such methods are usually formulated as attempting to solve a type of unsupervised learning problem known as clustering.

One popular subclass of clustering algorithms are partitional algorithms which divide a set of data points into K groups in such a way that maximizes the within group similarity. Algorithms in this class can generally be interpreted as attempting to optimize some cost function J that is related to the within group similarity. Many standard approaches to partitional clustering assume that associated with each cluster k is a cluster prototype μ_k. Given a measure of similarity between data points $S(\cdot,\cdot)$, the cost function can then be defined as the net similarity of data points to the prototype of their assigned cluster, which can be written as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} S(x_n, \mu_k) \tag{2.1}$$

where $r_{nk} = 1$ when x_n is assigned to cluster k and $r_{nj} = 0$ for $j \neq k$.

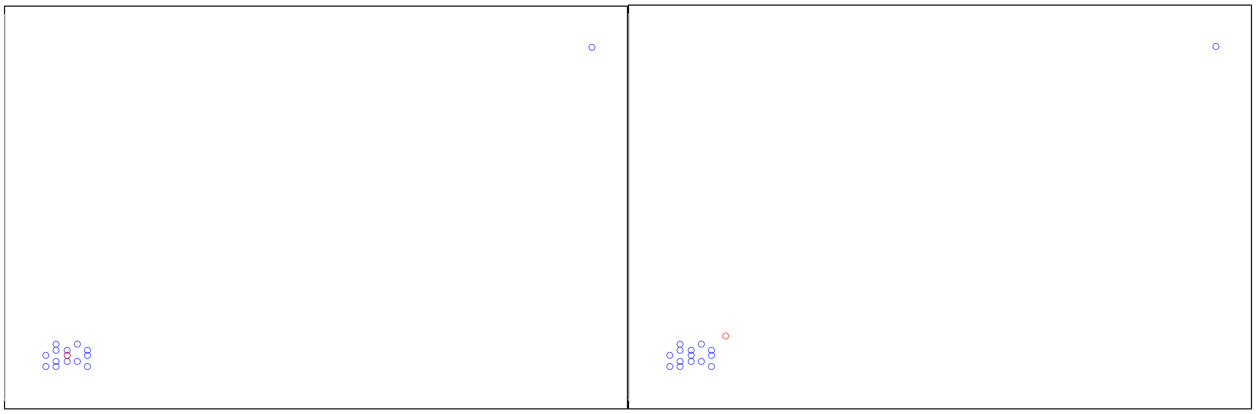
Some approaches, such as the K-means algorithm allow the cluster prototypes to be points not included in the original data set. However, there are several reason why it is preferable to restrict the cluster prototypes to only take on values equal to points from the original data set. One reason is that more flexibility is afforded in the similarity measure because it need only be defined for points in the original data set. Another reason is that restricting the values that can be taken by cluster prototypes increases the robustness of the cluster prototypes to outliers as demonstrated in Fig. 2-1. A third reason is that cluster prototypes can be used as candidates for further inspection after clustering. For example, in an application of clustering to genomic data, a cluster prototype that is a point from the data set will be a gene from the genome and can be examined in future experiments. Cluster prototypes that are restricted to only take values equal to points from the original data set will be referred to as exemplars in the rest of this discussion.

Optimizing J exactly is known to be NP-Hard [5]. However, several methods for finding an approximate solution have been developed. The K-medians algorithm is a popular algorithm that takes an iterative approach to the approximate optimization of J. The algorithm proceeds as follows. The μ_k are initially set to be equal to K arbitrary points from the data set. Then J is maximized with respect to the r_{nk}, keeping the μ_k fixed

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\ \min_j S(x_n, \mu_j) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Next J is maximized with respect to the μ_k, keeping the r_{nk} fixed. This is done by searching over the N_k data points currently in each cluster k. The two maximization steps are repeated iteratively until convergence.

The K-medians algorithm is used often in practice, yet it has several drawbacks. One important issue has to do with the convergence of the algorithm. Convergence of the al-



(a) (b)

Figure 2-1: The median of a set of points is more robust to outliers than the mean. (a) and (b) show the same set of points in blue. In (a), the median point is highlighted in red. This is a point that might be chosen as a prototype if prototypes are restricted to be points from the data set. In (b) the mean of the points is highlighted in red. This might be the point chosen as a prototype if prototypes are not constrained. In the unconstrained case, the prototype does not reflect the natural center of the cluster because of the outlying point.

gorithm is guaranteed, but only to a local maximum. Thus, in practice, the K-medians algorithm is generally run many times on the same data set with different initializations of the μ_k. The ability of the algorithm to find a good solution is very sensitive to how it is initialized, especially when the data set has many points and K is large. Another difficulty in using the K-medians algorithm effectively is that K must be specified when it is usually not known how many clusters are present in the data. In the next section, a partitional clustering algorithm called Affinity Propagation is discussed. Affinity Propagation approximately optimizes a cost function similar to J while addressing to some extent the issues that plague the K-medians algorithm.

2.2 Affinity Propagation

The approach to clustering taken by Affinity Propagation (AP) [10] is similar to the K-medians problem in that the objective is to maximize the similarity of points to their assigned exemplars. However, AP avoids the problem of having to choose an appropriate value for K

to some extent by incorporating the selection of the number of clusters into the optimization problem. This is accomplished by associating a cost with a point choosing itself as an exemplar and then enforcing that if any point x_i takes x_j to be its exemplar, then x_j must be its own exemplar. Thus, the cost function becomes:

$$J = \sum_{i=1}^{N} s_i(c_i) + \sum_{k=1}^{N} \delta_k(\mathbf{c}) \tag{2.3}$$

Where c_i is the assigned exemplar for x_i, $s_i(\cdot) = S(x_i, \cdot)$, and $\delta_k(\cdot)$ enforces the constraint on valid exemplars. It is defined as follows:

$$\delta_k(\mathbf{c}) = \begin{cases} -\infty & \text{if } c_k \neq k \text{ but } \exists i : c_i = k \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Thus, a configuration of the variables $\mathbf{c} = \{c_1, \ldots, c_N\}$ that maximizes J will maximize the sum of similarities between data points and "valid" exemplars. The number of valid exemplars that are included in a maximal configuration of $\mathbf{c}$ will depend on the self-similarity values $s_i(i)$. When $s_i(i)$ is more negative, the ith data point will be less likely to be chosen as an exemplar because there will be more of a penalty on the cost function for doing so. This is because the $\delta_k(\cdot)$ function enforces that $c_i = i$ in any valid configuration that includes the ith point as an exemplar.

To find an optimal configuration of $\mathbf{c}$, Frey and Dueck recognized that by exponentiating the cost function J, it can be interpreted as a probability distribution: $p(\mathbf{c}) \propto \exp(J)$. In other words, $p(\mathbf{c})$ is a likelihood function over the random variables $\mathbf{c}$ and J is equal to the log likelihood $\ln p(\mathbf{c})$ up to some constant. A variety of approximate inference methods have been developed for finding the most likely setting of variables in a probabilistic model and thus can be used to find a configuration of $\mathbf{c}$ that maximizes $p(\mathbf{c})$. This approach to optimization has been applied in the clustering setting previously. In fact, as mentioned earlier, the K-means algorithm can be thought of as an application of an approximate inference method called the EM algorithm to a special case of the Gaussian mixture model [3]. The approach that Frey and Dueck chose to take was to represent $p(\mathbf{c})$ as a graphical model, specifically a factor graph, in order to utilize the Max-Sum algorithm for inference [18]. To facilitate the discussion of AP, a brief introduction to factor graphs and the Max-Sum algorithm will be given.

2.2.1 Factor Graphs

A factor graph is a special type of graphical model formalism [15]. In general, graphical models are a useful methodology for representing probability distributions diagrammatically. Graphical models are used extensively in machine learning research and many important results have been derived using graphical models. Common variants of graphical models include Bayesian networks and Markov random fields in addition to factor graphs. Factor graphs are discussed in this section because they will provide a convenient framework for describing the Max-Sum algorithm.

Let $\mathbf{x}$ be the set of variables $\{x_1, \ldots, x_N\}$ and suppose that the joint distribution over $\mathbf{x}$ can be written as a product of functions (factors):

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) \tag{2.5}$$

where $\mathbf{x}_s$ denotes the subset of $\mathbf{x}$ whose members are the arguments to the function $f_s(\cdot)$.

In the factor graph representation of $p(\mathbf{x})$, a circular node is associated with each variable x_i and a square node is associated with each factor $f_s(\mathbf{x}_s)$. Undirected edges connect each factor node to the variable nodes on which that factor depends. As a more concrete example, consider the following factorization of a distribution

$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \tag{2.6}$$

The factor graph representing this factorization is shown in Fig. 2-2.

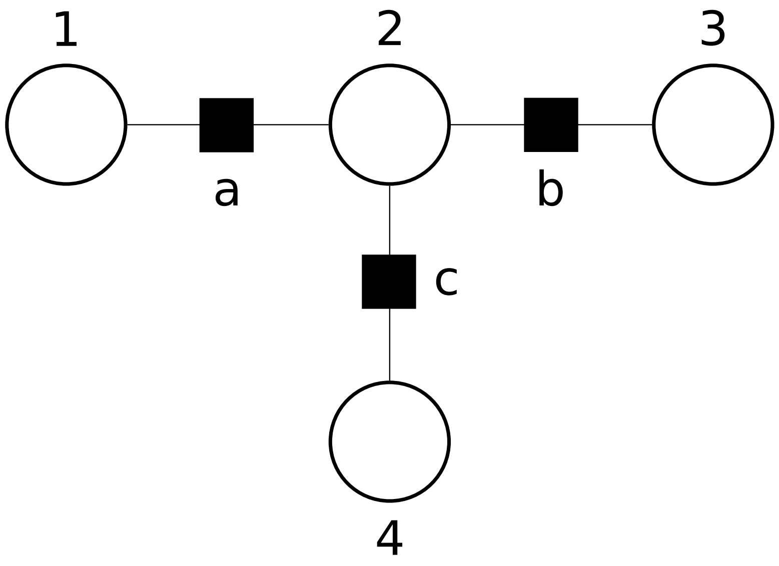


Figure 2-2: The factor graph representing the factorization in Eqn. 2.6. Circular nodes represent variables and square nodes represent factors. The node labels designate which variable or factor the node represents.

2.2.2 The Max-Sum Algorithm

A common task when working with probability distributions is to find a maximal setting of the variables. The max-sum algorithm is an algorithm for finding the maximum likelihood solution given a factor graph representation of a probability distribution. The algorithm will be described in this section. For a more thorough derivation see [3].

The max-sum algorithm operates by passing messages between nodes in a factor graph. The messages are functions of the variables represented by the variable nodes that are either the recipients or senders of the messages. The logarithms of the functions specified by the factor graph are evaluated during the course of the algorithm. This allows sums of messages to be evaluated rather than products, thus avoiding issues of numerical precision. The name "max-sum" reflects this aspect of the algorithm. Two types of messages are defined depending on whether the message is being passed to or from a factor node. In the following equations, the x_i are the variables over which the probability distribution is defined. The f_j are the functions that make up the factorization of the probability distribution as reflected in the factor graph.

variable node to factor node:

$$\mu_{x_i \to f_j}(x_i) = \sum_{f_k \in n(x_i) \setminus \{f_j\}} \mu_{f_k \to x_i}(x_i) \tag{2.7}$$

factor node to variable node:

$$\mu_{f_j \to x_i}(x_i) = \max_{\mathbf{x}_j \setminus \{x_i\}} \left[\ln f_j(\mathbf{x}_j) + \sum_{x_k \in \mathbf{x}_j \setminus \{x_i\}} \mu_{x_k \to f_j}(x_k) \right] \tag{2.8}$$

where $n(x_i)$ is the set of factor nodes that neighbor the variable node x_i in the factor graph.

When the max-sum algorithm is applied to a tree structured graph, outgoing messages are sent by a node when all of the necessary incoming messages have been received. Message passing is initiated by the leaf node and messages propagate through the graph. Message passing terminates when the leaf nodes receive messages from their neighbors.

To determine the maximal setting of each variable x_i once message passing has terminated, the arg max is taken over all incoming messages to the variable node

$$x_i^{\max} = \arg\max_{x_i} \left[\sum_{f_k \in n(x_i)} \mu_{f_k \to x_i}(x_i) \right] \tag{2.9}$$

The max-sum algorithm is an efficient and straight-forward algorithm for exact inference in tree-structured factor graphs. In general graphs with cycles, it is possible to apply a iterative version of the max-sum algorithm in which all nodes send messages at each iteration based on the current set of incoming messages. The theoretical properties of the max-sum algorithm and other related algorithms when applied to an arbitrary graph with cycles are not fully understood [26]. Although there are no guarantees of convergence, it is often the case that the messages converge after several iterations and that an approximate solution may be extracted using Eqn. 2.9. Several important results (turbo codes [2] being one notable example) can be interpreted as an application of the max-sum algorithm or one of several related message passing algorithms to a graphical model with cycles. In the next section, such an application is used to derive Affinity Propagation.

2.2.3 Derivation of Affinity Propagation

The factor graph representation of the interpretation of J as a probability distribution is shown in Fig. 2-3. there are two types of factor nodes in this graph: those corresponding to local $s_i(\cdot)$ functions and those corresponding to $\delta_k(\cdot)$ functions. We will therefore refer to those nodes as s nodes and δ nodes. Max-sum messages may be derived according to the message definitions given above. The messages from δ nodes to variable nodes are

$$\mu_{\delta_k \to c_i}(c_i) = \max_{\mathbf{c} \setminus \{c_i\}} \left[\delta_k(\mathbf{c}) + \sum_{c' \in \mathbf{c} \setminus \{c_i\}} \mu_{c' \to \delta_k}(c') \right] \tag{2.10}$$

The s nodes are each only connected to one variable node, for example the node corre-

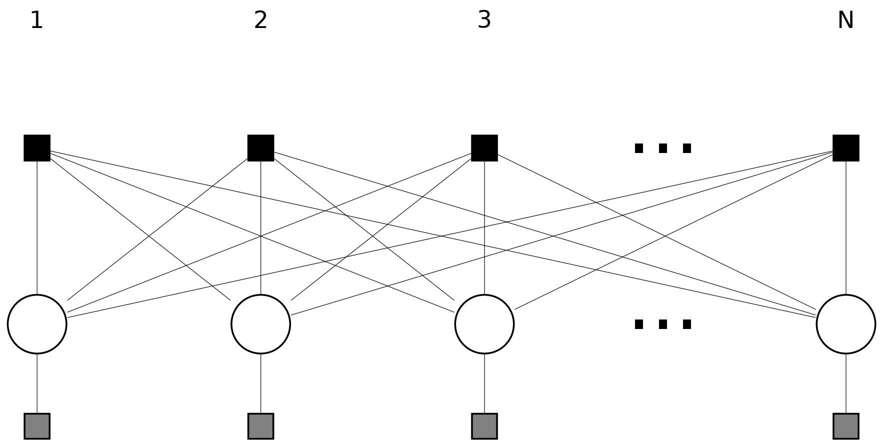


Figure 2-3: The factor graph representing the factorization of $p(\mathbf{c}) \propto \exp(J)$. The variables are represented by circular nodes. For each variable c_i there is a factor corresponding to the function $\delta_i(\mathbf{c})$ represented by a black square node and a factor corresponding to the function $s_i(c_i)$ represented by a gray square node.

sponding to the function $s_i(c_i)$ is connected to the node corresponding to the variable c_i. Thus, the messages from the s nodes to variable nodes simplify:

$$\mu_{s_i \to c_i}(c_i) = s_i(c_i) \tag{2.11}$$

The messages from variable nodes to δ nodes take the following form:

$$\mu_{c_i \to \delta_k}(c_i) = \mu_{s_i \to c_i}(c_i) + \sum_{k' \neq k} \mu_{\delta_{k'} \to c_i}(c_i) \tag{2.12}$$

The calculation of the simplified version of the messages from s nodes to c nodes does not involve any incoming messages, so there is no need to calculate messages from c nodes to s nodes.

One issue made apparent by the message definitions given above is that the $\delta_k(\cdot)$ functions take the entire vector $\mathbf{c}$ as an argument. Calculating the message from a δ node to a variable node involves finding a maximal configuration of $\mathbf{c}$. To do this as written would require searching over the space of configurations of $\mathbf{c}$ which is of size exponential in the number of variables. It would seem that calculating these messages is not any more computationally feasible than maximizing J directly. Fortunately, the way the $\delta_k(\cdot)$ functions are defined allows for a powerful simplification of the calculation of messages from δ nodes to variable nodes. Note that when the constraint imposed by the $\delta_k(\cdot)$ functions is violated the value of the entire expression is equal to $-\infty$. Thus there is no reason to even consider configurations of $\mathbf{c}$ that violate this constraint. The calculation of a message from a δ node to a variable node involves a maximization over all of the components of $\mathbf{c}$ except the component corresponding to the variable node to which the message is being sent. The message is a function of that component, so a value of the message must be calculated for each value that the component can take. The important observation is that for the message $\mu_{\delta_k \to c_i}(c_i)$ the constraint imposed by the $\delta_k(\cdot)$ function on the values of the components of $\mathbf{c}$ other than c_i is the same for all values of c_i except for $c_i = k$. What the constraint actually is depends on whether k equals i or not.

	$c_i = k$	$c_i \neq k$
$k = i$	point i is an exemplar for itself and may be an exemplar for other points	point i is not an exemplar for itself and may not be an exemplar for other points
$k \neq i$	point k must be an exemplar for itself and may be an exemplar for other points	point k may be an exemplar for itself in which case it may be an exemplar for other points or it may not be an exemplar for itself in which case it may not be an exemplar for other points

Table 2.1: The interpretation of the constraint imposed by the δ_k functions for different values of c_i and k.

Table 2.1 summarizes the effects that the values of c_i and k have on the constraint imposed by the $\delta_k(\cdot)$ functions. Notice that there are only two ways that the $\delta_k(\cdot)$ functions can constrain the configurations of $\mathbf{c}$ that are to be considered in the calculation of any given message from a δ node to a variable node. This observation has two important consequences. One is that the $\delta_k(\cdot)$ functions may be effectively disregarded and the max operators may be pushed through the sum and operate directly on the messages which are functions of individual components of $\mathbf{c}$. In order to disregard the $\delta_k(\cdot)$ functions, the max operators operate over appropriately constrained sets of values. Searching over the configurations of the components of $\mathbf{c}$ individually is much more efficient than searching over the entire configuration space of $\mathbf{c}$. The second consequence mentioned above is that the messages only range over two values, one when $c_i = k$ and one when $c_i \neq k$. Thus, in the implementation of AP, messages from δ nodes to variable nodes may be represented as pairs of values rather than vectors with length equal to the number of data points. The message may be expressed without the $\delta_k(\cdot)$ functions in the following way:

$$\mu_{\delta_k \to c_i}(c_i) = \tag{2.13}$$

$$\begin{cases} \sum_{c' \in \mathbf{c}\setminus\{c_i\}} \max_{c'} \mu_{c' \to \delta_k}(c') & c_i = k \text{ and } k = i \\ \sum_{c' \in \mathbf{c}\setminus\{c_i\}} \max_{c' \neq k} \mu_{c' \to \delta_k}(c') & c_i \neq k \text{ and } k = i \\ \mu_{c_k \to \delta_k}(k) + \sum_{c' \in \mathbf{c}\setminus\{c_i, c_k\}} \max_{c'} \mu_{c' \to \delta_k}(c') & c_i = k \text{ and } k \neq i \\ \max\left[\sum_{c' \in \mathbf{c}\setminus\{c_i\}} \max_{c' \neq k} \mu_{c' \to \delta_k}(c'), \mu_{c_k \to \delta_k}(k) + \sum_{c' \in \mathbf{c}\setminus\{c_i, c_k\}} \max_{c'} \mu_{c' \to \delta_k}(c')\right] & c_i \neq k \text{ and } k \neq i \end{cases}$$

Another issue with the max-sum algorithm besides the tractability of message calculation is the potential for numerical overflow. The calculation of each message involves a sum with a number of terms that is on the order of the number of data points. After several iterations of message-passing these messages may become quite large in magnitude. To avoid this situation in AP, the messages are rescaled after each iteration. This does not affect the solution obtained by the max-sum algorithm if it converges because scaling each message by a factor that is independent of the argument to the message does not change the set of arguments that maximize the sum of the messages that are sent to a variable node at a particular iteration. Frey and Dueck suggest a scaling method that is convenient to implement and efficient. Messages from variable nodes to δ nodes are scaled by the maximum value that they achieve over argument values other than the index of the δ node to which the message is being sent:

$$\tilde{\mu}_{c_i \to \delta_k}(c_i) = \mu_{c_i \to \delta_k}(c_i) - \max_{c_i \neq k} \mu_{c_i \to \delta_k}(c_i) \tag{2.14}$$

This scaling method is convenient because the following equalities now hold:

$$\begin{aligned} \max_{c_i \neq k} \tilde{\mu}_{c_i \to \delta_k}(c_i) &= 0 && (2.15) \\ \max_{c_i} \tilde{\mu}_{c_i \to \delta_k}(c_i) &= \max\left[0, \tilde{\mu}_{c_i \to \delta_k}(k)\right] && (2.16) \end{aligned}$$

Messages from δ nodes to variable nodes, which range over two values, are scaled by the value that they take when $c_i \neq k$. Using the equalities derived directly above, the scaled messages from δ nodes to variable nodes can be expressed as

$$\tilde{\mu}_{\delta_k \to c_i}(c_i) = \tag{2.17}$$

$$\begin{cases} \sum_{c' \in \mathbf{c} \setminus \{c_i\}} \max\left[0, \tilde{\mu}_{c' \to \delta_k}(k)\right] & c_i = k \text{ and } k = i \\ 0 & c_i \neq k \text{ and } k = i \\ \min\left[\tilde{\mu}_{c_k \to \delta_k}(k) + \sum_{c' \in \mathbf{c} \setminus \{c_i, c_k\}} \max\left[0, \tilde{\mu}_{c' \to \delta_k}(k)\right], 0\right] & c_i = k \text{ and } k \neq i \\ 0 & c_i \neq k \text{ and } k \neq i \end{cases}$$

Given this expression for the scaled messages from δ nodes to variable nodes, the scaled messages from variable nodes to δ nodes can be expressed in the following way. Note that the message $\tilde{\mu}_{c_i \to \delta_k}(c_i)$ is only evaluated for $c_i = k$ in the calculation of the scaled messages from δ nodes to variable nodes. Thus, an expression for the scaled messages from variable nodes to δ nodes is only needed for the case when $c_i = k$.

$$\begin{aligned} \tilde{\mu}_{c_i \to \delta_k}(k) &= \tilde{\mu}_{s_i \to c_i}(k) + \sum_{k' \neq k} \tilde{\mu}_{\delta_{k'} \to c_i}(k) - \max_{c_i \neq k}\left[\tilde{\mu}_{s_i \to c_i}(c_i) + \sum_{k' \neq k} \tilde{\mu}_{\delta_{k'} \to c_i}(c_i)\right] \\ &= \tilde{\mu}_{s_i \to c_i}(k) - \max_{c_i \neq k}\left[\tilde{\mu}_{s_i \to c_i}(c_i) + \tilde{\mu}_{\delta_{c_i} \to c_i}(c_i)\right] \end{aligned}$$

As a result of this rescaling method, messages going both directions between δ nodes and variable nodes have been reduced to scalar values that can be computed in $O(N)$ time where N is the number of data points. Each iteration of AP takes $O(N^2)$ time because $O(N)$ messages are sent at each iteration.

Rescaling the messages seems to allow message-passing to proceed indefinitely. The possibility that the messages will not converge is still a concern given the cyclic nature of the factor graph. One common cause of non-convergence for the AP messages is the presence of data points that are are almost equally viable as exemplars for each other. This situation often results in messages that oscillate during the course of message-passing. Frey and Dueck incorporate a “damping factor” λ to avoid oscillation. So, in AP, the messages after each iteration are linear combinations of λ times the old value of the message plus $(1-\lambda)$ times what the new value of the message should be according to the scaled update rules defined in the expressions above. Frey and Dueck claim (but do not prove) that they are always able to avoid non-convergence due to oscillation by increasing λ [10].

To extract a solution from converged messages, Frey and Dueck employ a non-standard approach. Rather than use all of the exemplars specified by a maximal configuration of $\mathbf{c}$, they decided to use only “strong” exemplars. In other words, in a maximal configuration of $\mathbf{c}$ there might be a point chosen to be an exemplar for which J would not change greatly if another point were chosen to be an exemplar instead. Such a point Frey and Dueck would not take as an exemplar because it might not be very informative. As an approximate method for finding “strong” exemplars, a point k is chosen as an exemplar if

$$\tilde{\mu}_{\delta_k \to c_k}(k) + \tilde{\mu}_{c_k \to \delta_k}(k) > 0 \tag{2.18}$$

If the set of exemplars found using the above method remains the same after several message-passing iterations, the messages are considered to have converged. Points are assigned to the exemplar to which they are the most similar. Because only “strong” exemplars are considered, potentially fewer exemplars are included in the results than if “weak” exemplars were not filtered out of the results found with the max-sum algorithm.

2.3 Time Series Affinity Propagation

AP has been shown to perform very well at clustering many different types of data. In the case of time-series data there are at least two different ways that AP could be applied to discover clusters. One way would be to concatenate the data from all time points for each object and thus assign each object to one cluster. This approach would lose information about the dynamics of the data because it would not be possible for objects to belong to different clusters at different time points. Another approach would be to cluster each object separately at each time point. This approach would recover the trajectory through clusters that each object takes over time. However, an assumption is made that the behavior of objects at different time points is independent. In many situations this is an invalid assumption because the behavior of an object at one point in time will be related to its behavior at recent points in time. The following discussion introduces a clustering method related to AP in which the assignment of data points to clusters at each time point is made separately but is influenced by the assignments made to clusters at other time points. This method will be referred to as Time Series Affinity Propagation (TSAP).

To model the manner in which objects behave over time J is modified in the following manner.

$$\tilde{J} = \sum_{i=1}^{N}\sum_{j=1}^{T} s_{i,j}(c_{i,j}) + \sum_{i=1}^{N}\sum_{j=1}^{T} \delta_{i,j}(\mathbf{c}) + \gamma(\mathbf{c}) \tag{2.19}$$

The vector $\mathbf{c}$ is now indexed by object in the dataset and by time point. The $s_{i,j}(\cdot)$ and $\delta_{i,j}(\cdot)$ functions have definitions that are analogous to their definitions in the original cost function. The $\gamma(\cdot)$ function can be defined in various ways in order to influence the value of the cost function. The particular way in which the $\gamma(\cdot)$ function is defined depends on the relationship among the components of $\mathbf{c}$ that is to be modeled.

The approach taken in this thesis to model the temporal relationship among data points in time series data is to define the $\gamma(\cdot)$ function such that the cost function is influenced by the way that data points cluster at consecutive time points. This essentially assumes a first order Markov property for the trajectories. The $\gamma(\cdot)$ function to be defined as a sum of functions $\gamma_{i,j}(\cdot,\cdot)$

$$\tilde{J} = \sum_{i=1}^{N}\sum_{j=1}^{T} s_{i,j}(c_{i,j}) + \sum_{i=1}^{N}\sum_{j=1}^{T} \delta_{i,j}(\mathbf{c}) + \sum_{i=1}^{N}\sum_{j=1}^{T-1} \gamma_{i,j}(c_{i,j}, c_{i,j+1}) \tag{2.20}$$

To fully constrain data points at consecutive time points to belong to the same cluster, the γ factors would be defined as

$$\gamma_{i,j}(c_{i,j}, c_{i,j+1}) = \begin{cases} 0 & c_{i,j} = c_{i,j+1} \\ -\infty & \text{otherwise} \end{cases} \tag{2.21}$$

This definition of the $\gamma_{i,j}(\cdot,\cdot)$ factors can be generalized to allow a variable degree of influence between the manner in which data points at consecutive time points cluster.

$$\gamma_{i,j}(c_{i,j}, c_{i,j+1}) = \begin{cases} \alpha_1 & c_{i,j} = c_{i,j+1} \\ \alpha_2 & \text{otherwise} \end{cases} \tag{2.22}$$

To maximize $\tilde{J}$, the max-sum algorithm can be employed. The factor graph representation of $\tilde{J}$ is shown in Fig. 2-4. The messages from s nodes to variable nodes are essentially the same and can be simplified to be

$$\mu_{s_{i,j} \to c_{i,j}}(c_{i,j}) = s_{i,j}(c_{i,j}) \tag{2.23}$$

The messages from δ nodes to variable nodes are also essentially the same and through the same scaling trick can be expressed as

$$\tilde{\mu}_{\delta_{k,l} \to c_{i,j}}(c_{i,j}) = \tag{2.24}$$

$$\begin{cases} \sum_{c' \in \mathbf{c} \setminus \{c_{i,j}\}} \max\left[0, \tilde{\mu}_{c' \to \delta_{k,l}}(<k,l>)\right] & c_{i,j} =<k,l> \text{ and } <k,l>=<i,j> \\ 0 & c_{i,j} \neq <k,l> \text{ and } <k,l>=<i,j> \\ \min\left[\tilde{\mu}_{c_{k,l} \to \delta_{k,l}}(<k,l>) + \right. & \\ \left. \sum_{c' \in \mathbf{c} \setminus \{c_{i,j}, c_{k,l}\}} \max\left[0, \tilde{\mu}_{c' \to \delta_{k,l}}(<k,l>)\right], 0\right] & c_{i,j} =<k,l> \text{ and } <k,l>\neq<i,j> \\ 0 & c_{i,j} \neq <k,l> \text{ and } <k,l>\neq<i,j> \end{cases}$$

The messages from γ nodes to variable nodes are as follows

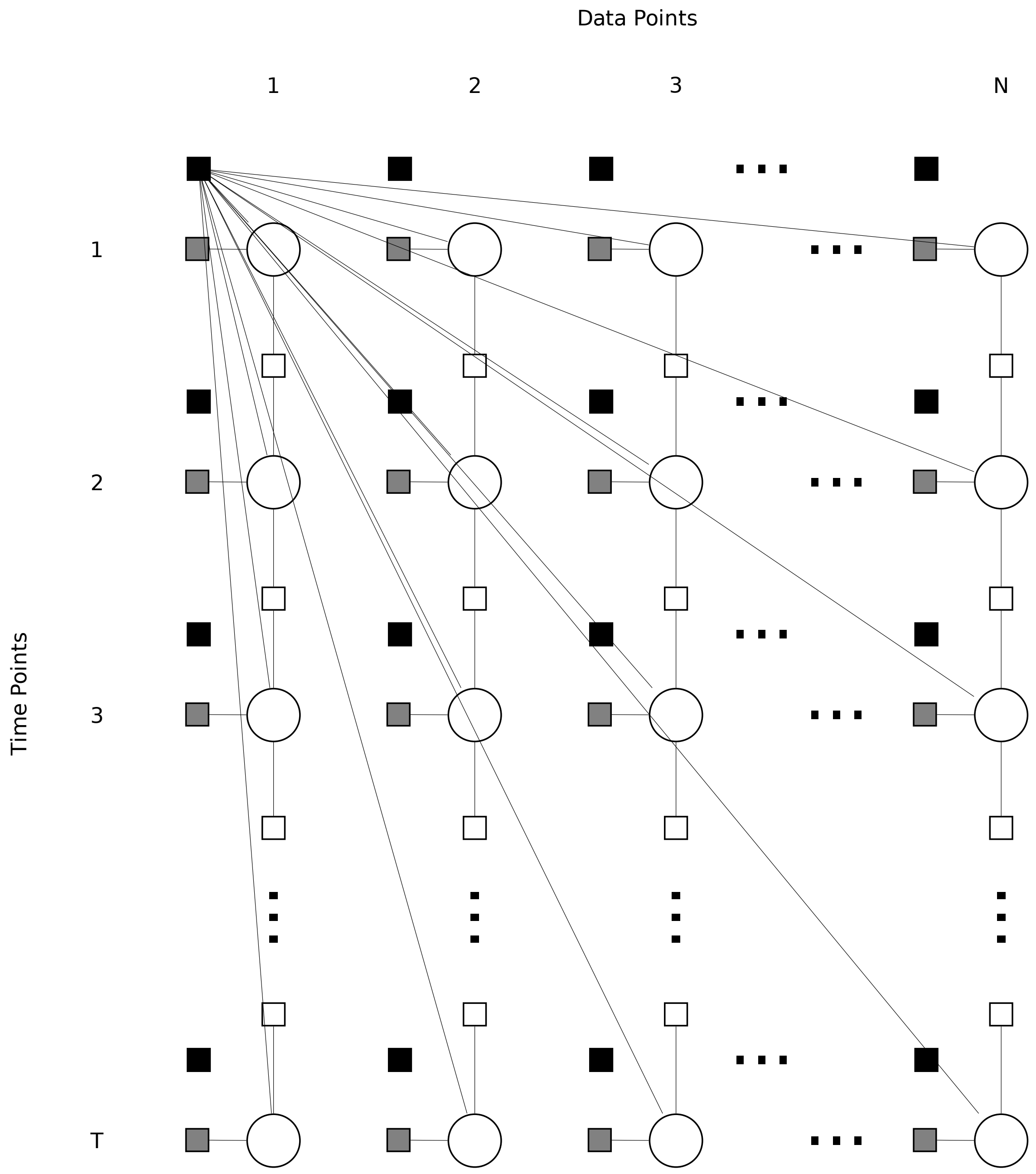


Figure 2-4: The factor graph representing the factorization of $p(\mathbf{c}) \propto \exp(\tilde{J})$. The variables are represented by circular nodes. For each variable $c_{i,j}$ there is a factor corresponding to the function $\delta_{i,j}(\mathbf{c})$ represented by a black square node and a factor corresponding to the function $s_{i.j}(c_{i.j})$ represented by a gray square node. Between each pair of variables $c_{i,j}$ and $c_{i,j+1}$ there is a factor corresponding to the function $\gamma_{i,j}(c_{i,j}, c_{i,j+1})$ represented by a white square node. Each δ node is connected by an edge to every variable node. Only the edges connected to the node representing the factor corresponding to the function $\delta_{1,1}(\mathbf{c})$ are displayed for clarity.

$$\begin{aligned}
\mu_{\gamma_{i,j}\to c_{i,j}}(c_{i,j}) &= \max_{c_{i,j+1}} \left[\gamma_{i,j}(c_{i,j}, c_{i,j+1}) + \mu_{c_{i,j+1}\to\gamma_{i,j}}(c_{i,j+1})\right] && (2.25)\\
&= \max\left[\alpha_1 + \mu_{c_{i,j+1}\to\gamma_{i,j}}(c_{i,j}), \alpha_2 + \max_{c_{i,j+1}\neq c_{i,j}} \mu_{c_{i,j+1}\to\gamma_{i,j}}(c_{i,j+1})\right] && (2.26)
\end{aligned}$$

These messages can be calculated in $O(NT)$ time because the max over incoming messages only needs to be calculated once per iteration. Once again, the messages from variable nodes to s nodes do not need to be calculated because they are never used by any other calculation. The messages from variable nodes to δ nodes are the same as in standard AP with the addition of one or two incoming messages from γ nodes. They can be scaled in the same way and need only be calculated for $c_{i,j} =< k, l >$.

$$\begin{aligned}
\tilde{\mu}_{c_{i,j}\to\delta_{k,l}}(< k, l >) &= \tilde{\mu}_{s_{i,j}\to c_{i,j}}(< k, l >) + \mu_{\gamma_{i,j-1}\to c_{i,j}}(< k, l >) + \mu_{\gamma_{i,j}\to c_{i,j}}(< k, l >) + \\
&\quad \sum_{<k',l'>\neq<k,l>} \tilde{\mu}_{\delta_{k',l'}\to c_{i,j}}(< k, l >) - \\
&\quad \max_{c_{i,j}\neq<k,l>} \Big[\tilde{\mu}_{s_{i,j}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j-1}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j}\to c_{i,j}}(c_{i,j}) + \\
&\quad \sum_{<k',l'>\neq<k,l>} \tilde{\mu}_{\delta_{k',l'}\to c_{i,j}}(c_{i,j})\Big] \\
&= \tilde{\mu}_{s_{i,j}\to c_{i,j}}(< k, l >) + \mu_{\gamma_{i,j-1}\to c_{i,j}}(< k, l >) + \mu_{\gamma_{i,j}\to c_{i,j}}(< k, l >) - \\
&\quad \max_{c_{i,j}\neq<k,l>} \left[\tilde{\mu}_{s_{i,j}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j-1}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j}\to c_{i,j}}(c_{i,j}) + \tilde{\mu}_{\delta_{c_{i,j}}\to c_{i,j}}(c_{i,j})\right]
\end{aligned}$$

There is no obvious way to avoid calculating the messages from variable nodes to γ nodes for all values of $c_{i,j}$. However, because of the way that the messages from δ nodes to variable nodes are scaled, the messages from variable nodes to γ nodes can be calculated efficiently.

$$\begin{aligned}\mu_{c_{i,j}\to\gamma_{i,j}}(c_{i,j}) &= \mu_{s_{i,j}\to c_{i,j}}(c_{i,j}) + \sum_{n=1}^{N}\sum_{t=1}^{T}\mu_{\delta_{n,t}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j+1}\to c_{i,j}}(c_{i,j}) && (2.27)\\ &= \mu_{s_{i,j}\to c_{i,j}}(c_{i,j}) + \mu_{\delta_{c_{i,j}}\to c_{i,j}}(c_{i,j}) + \mu_{\gamma_{i,j+1}\to c_{i,j}}(c_{i,j}) && (2.28)\end{aligned}$$

For the messages to and from γ nodes, the versions for the variable node corresponding to the earlier time-point have been expressed above. The messages between a γ node and a later time-point are essentially the same except some variation in indexing. These messages should be scaled to avoid numerical overflow. In the absence of a scaling method that improves the efficiency of the calculation of any messages, the messages to and from γ node can be scaled by their maximum value.

To maximize the new cost function, the messages defined above are sent iteratively with the inclusion of a damping factor to avoid oscillations. The method of extracting "strong" exemplars that was used for AP cannot be used for this model because the selection of exemplars is influenced by the temporal constraint The arg max is taken over all incoming messages to each variable node to find the current approximation of the maximizing configuration. Once the approximation of the maximizing configuration has not changed for several iterations, this configuration is taken as the solution.

2.4 Similarity Measure Construction

The choice of similarity measure can have a drastic effect on the outcome of a clustering procedure. The similarity measure should reflect similarities and differences that are meaningful to the domain in which clustering is being applied. Similarity measure construction is made more complicated when the data come from heterogeneous sources. This section discusses an approach to incorporating expression as well as histone PTM data into a measure of similarity between genes.

To incorporate data from several sources, the information associated with a gene from each source is summarized as a scalar valued feature for that gene. The distributions of the features over all genes are then standardized so that the features corresponding to different data sources are on approximately the same scale. The set of feature vectors obtained in this way may be used to construct a similarity measure by taking the opposite of the Euclidean distance between each pair of vectors. The remainder of this section will discuss specific concerns with respect to extracting features from gene expression and histone PTM data as well as the appropriate distance measure to use to compare vectors of these features.

In the following it is assumed that there is a set X of N genes for which gene expression and histone PTM measurements have been taken at T time points. Self similarity values (e.g. $S(i,i)$) are left undefined because these values can be adjusted in order to make certain points more or less likely to be chosen as exemplars.

2.4.1 Gene Expression Data

Data collected using DNA microarray technology should be corrected for background noise using a method such as GCRMA [25] which uses a global model for the distribution of probe intensities and takes into account the different propensities of the probes to undergo non-specific binding based on their GC content. To make measurements from corresponding probes on arrays from different experiments comparable, a normalization procedure should also be applied. The quantile normalization method [4] corrects for between array noise by imposing the same empirical distribution of probe intensities to each array.

Following background adjustment and normalization, measurements from the same probe on different arrays are assumed to be comparable. It cannot be assumed that measurements from different probes are comparable even on the same array. For example, genes that are known to not be expressed in a given cell type by quantitative PCR exhibit probe measurements that differ by significant amounts. The cause of these discrepancies seems to be mainly based on sequence-specific effects of different probes. To account for between probe noise, measurements for each probe from the first time point are taken to be baseline measurements with which to normalize measurements from later time points. Unfortunately, this removes the first time point from direct consideration in the state inference procedure.

See [12] for a more thorough discussion of concerns related to the preprocessing of DNA microarray data.

2.4.2 Histone PTM Data

The histone PTM data that will be considered in this thesis are measured by chromatin immunoprecipitation combined with tiling array technology. Experimental evidence suggests that the appearance of histone PTMs around the transcription start sites (TSS) of genes is often indicative of gene regulation [17], so it is assumed that the tiling array includes probes at a high enough resolution such that at least 10-20 probes fall within a set window (e.g. 4000 base pairs) around the TSSs of the genes that are to be included in the analysis. The tiling array experiments should be conducted such that two channels of data may be extracted. One channel should be measured intensities corresponding to DNA fragments enriched by immunoprecipitation that have been hybridized to the array. The other channel should be measured intensities corresponding to DNA fragments that were not filtered by immunoprecipitation. To correct for noise between the two channels, they should be normalized in the following way. If ip and wce refer to the set of intensities from each channel, the ip channel should be normalized to be $ip' = ip \cdot \frac{median(wce)}{median(ip)}$. The $\frac{ip'}{wce}$ ratios can then be used as a quantitative measure of how enriched locations in the genome are for a histone PTM in the sample of cells used in the experiment. As with the microarrays used to measure expression, between array noise must be corrected. The quantile normalization method is also used for this purpose. An example of data that have been processed as described is shown in Fig. 2-5.

To extract a feature from the histone PTM data for each gene, the normalized ratios that fall within a set window around the TSS for each gene are considered as a vector. Because the feature extracted for each gene from the expression data represents an amount of change relative to the first time point, the features extracted from the histone PTM data should also represent change rather than an absolute value. The vector of ratios for the first time point is subtracted from the vectors from the later time points. The mean of the resulting vector

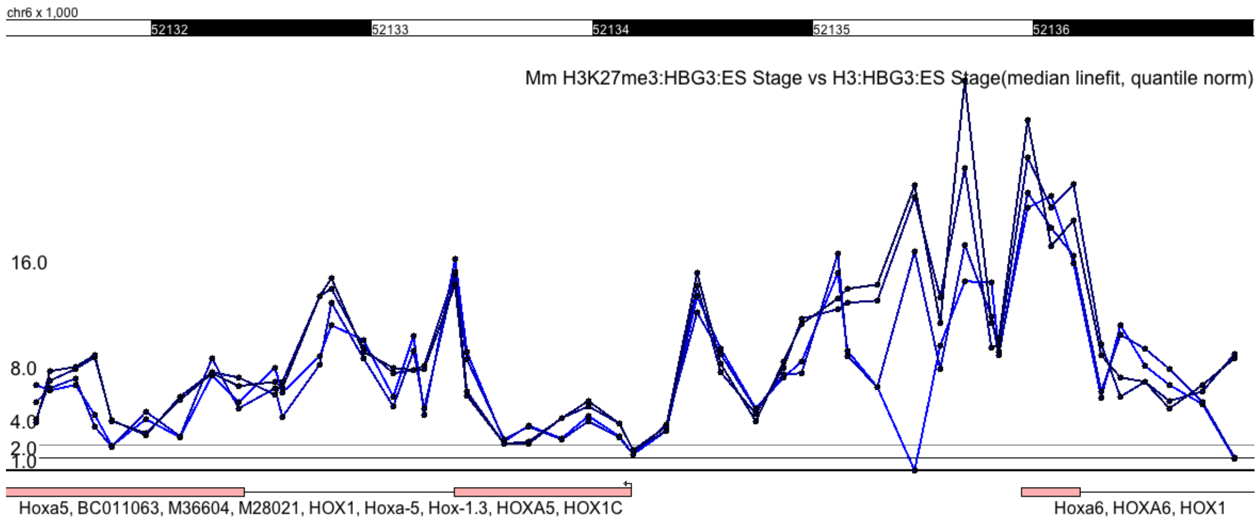


Figure 2-5: Data for the histone PTM H3K27me3 around the transcription start site of the gene Hoxa5 in mouse embryonic stem cells. Each location for which there are data points along the x-axis corresponds to a probe on the microarray. Multiple data points at each location reflect replicate experiments. Data points from the same replicate are connected by lines. The data have been normalized and thus the height of each data point is the normalized $\frac{ip'}{wce}$ ratio measured in the experiment.

of differences for each gene at each time point is taken as a feature for the construction of the similarity measure. If the ratios within the window around the TSS are thought of as being sampled points from a curve, the method of calculating a histone PTM feature as described above can be thought of as approximating the area between the curve from the time point for which the feature is being calculated and the curve from the first time point.

2.4.3 Comparing Feature Vectors

Once scalar valued features have been calculated for each gene at each time point other than the first time point, the similarity measure can be constructed by comparing the resulting vectors of features. If these vectors are considered to be points in a Euclidean space, Euclidean distance is a reasonable metric to use to compare points. To convert a distance measure to a similarity measure, the opposite of the distance measure is taken.

2.5 Visualizing States and Trajectories

Given a similarity measure defined over N genes at T time points, TSAP can be run to discover a set of states and the trajectories that genes take through them. To aid in the interpretation of results from TSAP, a visualization method is proposed in this section. State diagrams are abstractions used in systems research and theoretical computer science to represent simple machines. In its most basic form, a state diagram represents a set of states and the transitions that may be made between them as a graph. In the diagram, nodes represent states and edges represent possible transitions. This type of diagram is useful for visualizing the states found by TSAP and the transitions that genes take between states in their trajectories. Additional information may be conveyed through such diagrams by weighting nodes according to how often the corresponding states are visiting by the trajectories found by TSAP. Also, edges can be weighted according to how often the corresponding transitions are taken by the trajectories found by TSAP. An example state diagram is shown in Fig 2-6.

State diagrams compress the information about when during the time series states are visited and transitions are taken. Thus, a related form of expanded state diagram is used in the rest of this thesis. To preserve temporal information, states are drawn separately for each time point. The nodes can then be weighted according to how many trajectories pass through each state at each time point. Likewise, transitions made between each pair of consecutive time points can be drawn separately. An example of this type of state diagram is shown in Fig. 2-7.

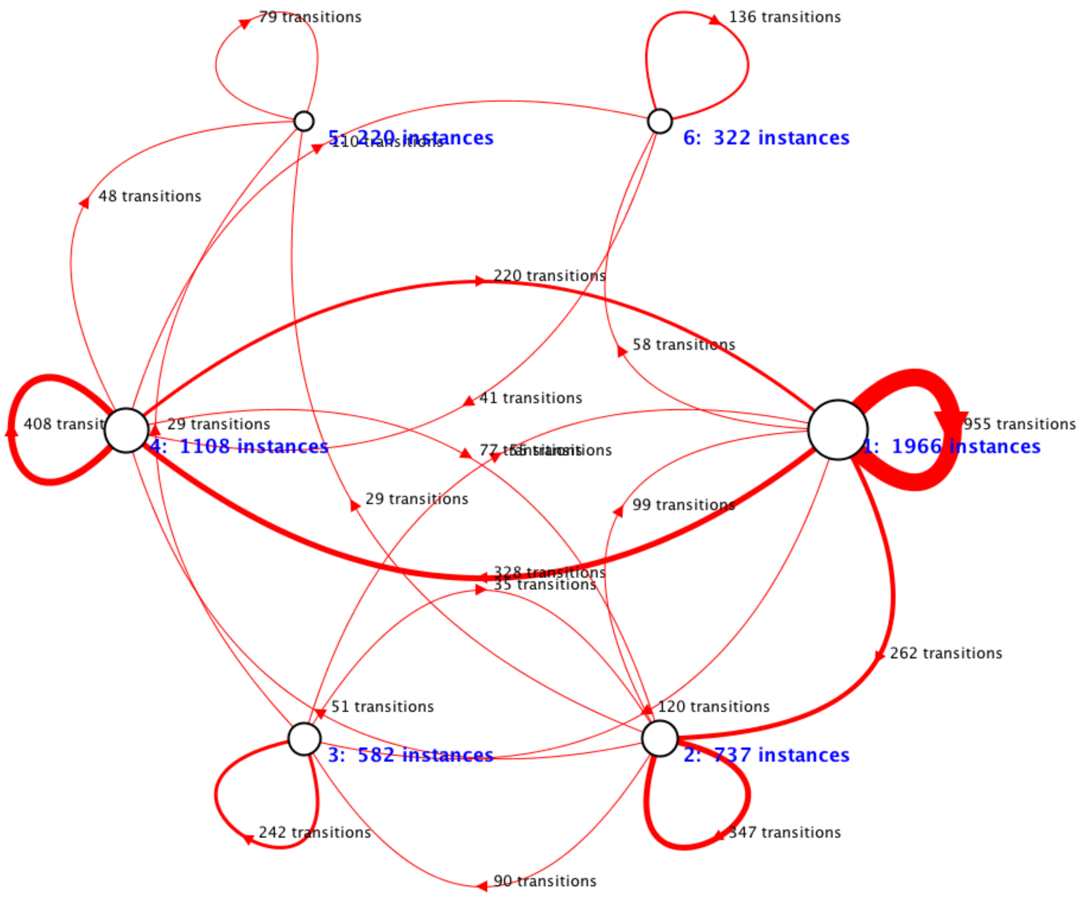


Figure 2-6: An example of a state diagram with six states. Nodes and edges are weighted to convey information about the usage of states and transitions by trajectories.

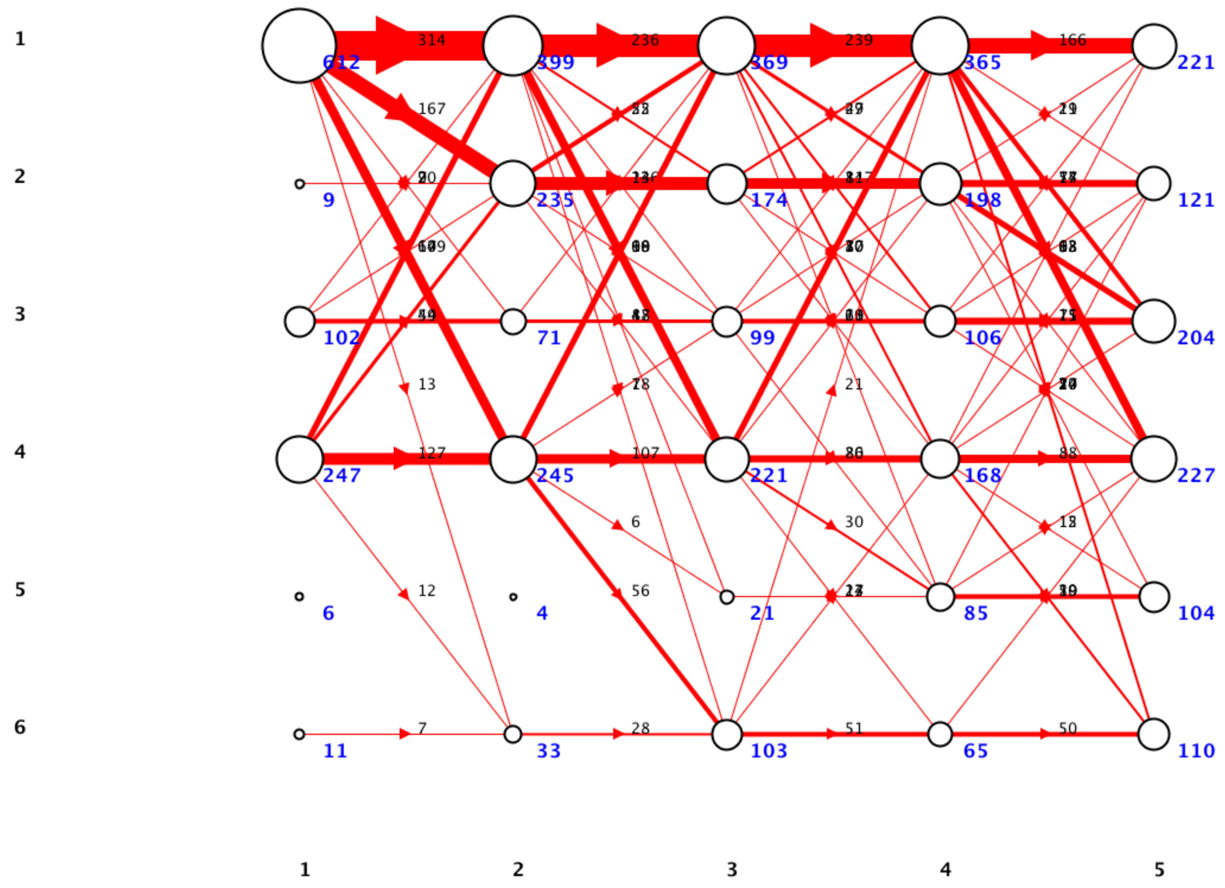


Figure 2-7: An example of an expanded state diagram with six states and five time points. Nodes and edges are weighted to convey information about the usage of states and transitions by trajectories. Edges for transitions taken by a small number of trajectories are left out to improve clarity.

Chapter 3

Applications

Several applications of TSAP are presented in this chapter. TSAP is initially applied to synthetic data to demonstrate its performance in a context where certain results are expected. An application to a real data set representing the development of a neural system in mouse is then presented. State diagrams are used to visualize the results from the real data. Several observations are made with respect to the analysis of the real data that are of biological significance.

In the applications of TSAP that follow, the self-similarity values $s_i(c_i)$ are always set to be the same in order to make all data points equally likely to be chosen as exemplars. To be consistent with the terminology of Frey and Dueck, the value that the self-similarity values are set to will be referred to as the preference value. Also, in examining various amounts of influence introduced by the $\gamma(\cdot, \cdot)$ function defined in Eqn. 2.22, the value of α_2 will be varied and the value of α_1 will always be set to 0 to reduce the complexity of the analysis. This is a reasonable decision because it makes for a more conservative analysis to assume that false transitions between states should be avoided rather than false transitions that stay in the same state. In the rest of this discussion, the value of α_2 will be referred to as the alpha value.

3.1 Synthetic Data

To demonstrate that TSAP is able to successfully recover states and trajectories from noisy data sets, the algorithm was applied to two sets of synthetic data. In the first set, data points were embedded in a two dimensional similarity space. This data set was designed to demonstrate TSAPs ability to constrain the clustering of temporally related points. The second set of synthetic data was designed to reflect patterns that are observed empirically in genomic time-series data. The performance of TSAP on biologically motivated synthetic data is supportive of the application of the algorithm to real data in the following section.

3.1.1 Toy Synthetic Data

A synthetic data set was constructed to contain points drawn from two Gaussian distributions with similar means and the same variance. One approach to attempting to recover which of the two distributions each point came from would be to partition the points into two clusters. The two clusters found by a clustering method such as AP would reflect the two underlying distributions in that the majority of the points assigned to a cluster would have been generated by one of the two distributions. However, many points would not be assigned to the cluster associated with the distribution that generated them. The reason for this is that many points will be closer to the center of the cluster associated with the wrong distribution because of the proximity of the means of the underlying distributions. A clustering method such as AP will assign points to the cluster with the exemplar to which the points are the most similar. As the results of TSAP on this synthetic data set will show, by incorporating a temporal constraint it is possible to assign more points to the cluster associated with the correct distribution.

The two distributions used to construct the synthetic data set were two-dimensional Gaussian distributions with means $(3, 2)$ and $(3, 4)$ and covariance matrix equal to the identity matrix. 100 points were sampled from each distribution. The points from each distribution were broken up into pairs so that the data set can be interpreted as observations of objects at two time points where the observations for a given object are from the same distribution at both time points. The constructed data set is shown in Fig. 3-1.

A similarity measure was constructed for this data set by taking the negative Euclidean distance between pairs of points. AP was run using this similarity measure over a range of preference values. Fig. 3-2 depicts the number of clusters found by AP over a range of preference values. The range of preference values for which AP finds two clusters is approximately -92 to -44. The performance of AP on this data set was evaluated by associating each of the exemplars from the runs that found two clusters to the distribution with the closest mean. An object was determined to be correctly assigned to a cluster if it is assigned to the cluster associated with the correct distribution. For the runs of AP that found 2 clusters, between 158 and 161 of the 200 points were assigned to the correct cluster. There is slight variation in the number of correctly clustered points because AP finds slightly different exemplars for different preference values, even when the same number of exemplars are found. Thus, about 20% of the points are closer to the wrong exemplar and AP cannot distinguish them from the points from the other distribution.

TSAP was also run on this data set for a range of preference and alpha values. The number of clusters found by TSAP varies with the parameter values (Fig. 3-3). The relationship between the preference value and the number of clusters found by TSAP is very similar to AP. When the alpha value is set to 0 and TSAP finds two clusters, the number of points that are correctly clustered ranges between 158 and 164. This range is somewhat different from the range found by AP because of the different manner in which points are assigned to clusters after convergence. The relationship between the alpha value and the mean number of correctly clustered points when TSAP finds two clusters is shown in Fig 3-4. The number of correctly clustered points increases as the alpha value becomes more negative and plateaus at -1.5. This is the alpha value less than which all objects are assigned to the same exemplar at both time points when TSAP finds two exemplars. In other words, the clustering of the objects is fully constrained when the alpha value is less than -1.5. By incorporating a constraint on the manner in which points cluster, TSAP is able to assign more points to the correct cluster. Some points are still clustered incorrectly at any alpha

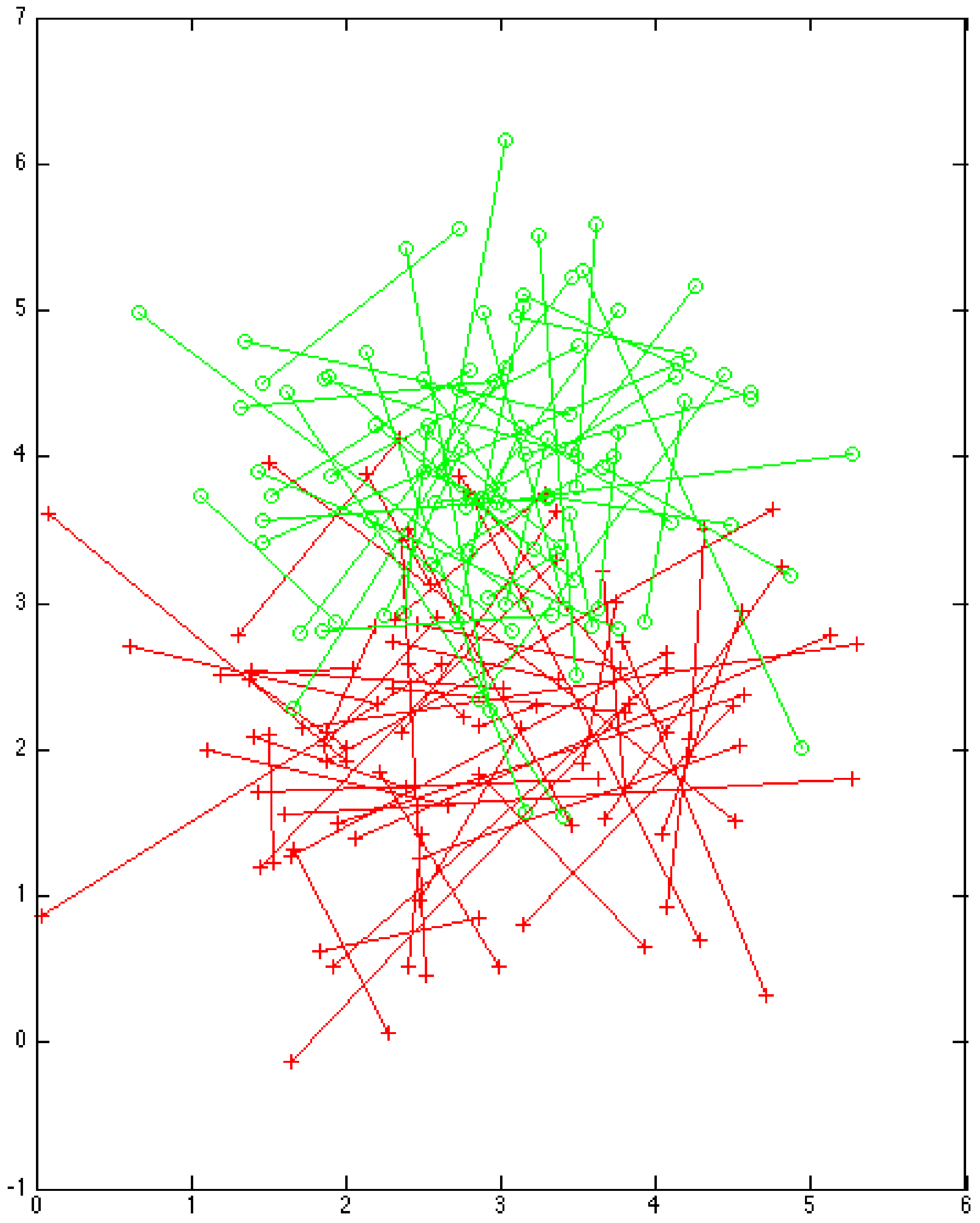


Figure 3-1: The synthetic data set consisting of points drawn from Gaussian distributions centered around $(3, 2)$ and $(3, 4)$. The points drawn from the two different distributions can be distinguished by their shape and color. Points corresponding to observations for the same object at different time points are connected by a line. In total, 50 objects were observed from each distribution at two time points.

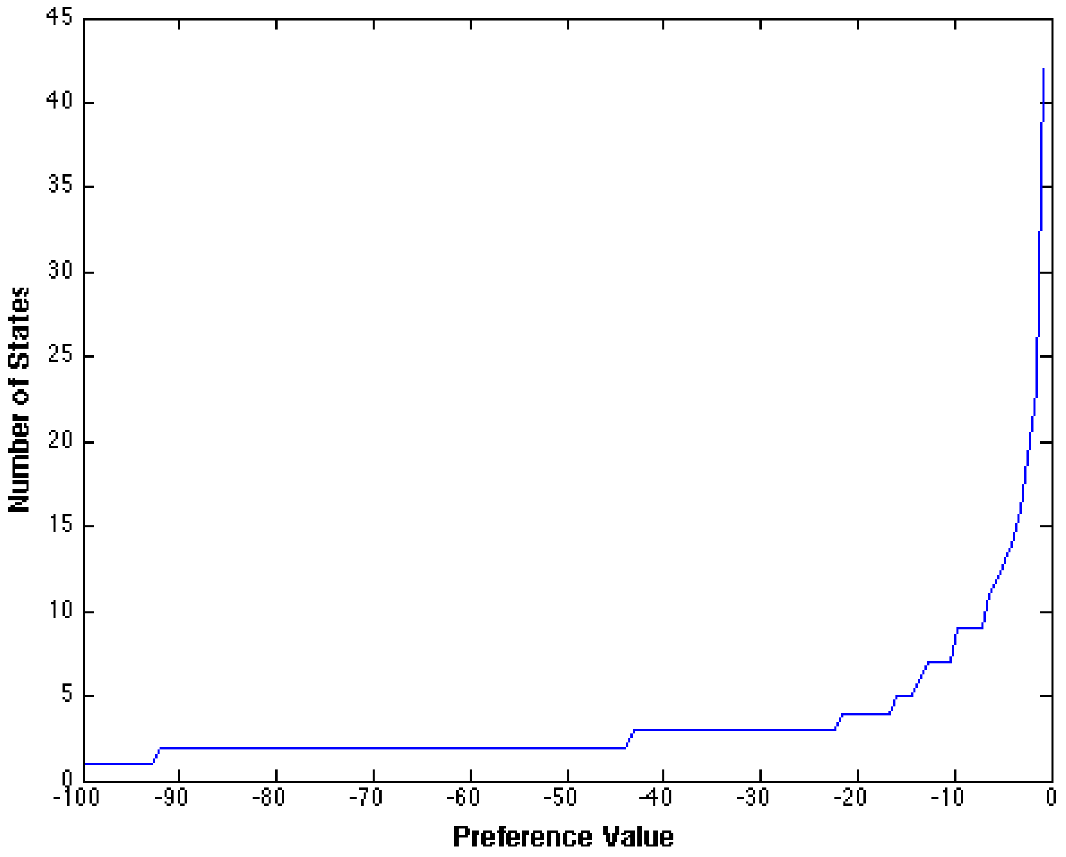


Figure 3-2: The number of clusters found by AP increases with the preference value.

value. This is either because both points in a connected pair happen to be similar to the exemplar of the incorrect exemplar or one of the two points is so similar to the incorrect exemplar that the other point is not able to influence its cluster assignment.

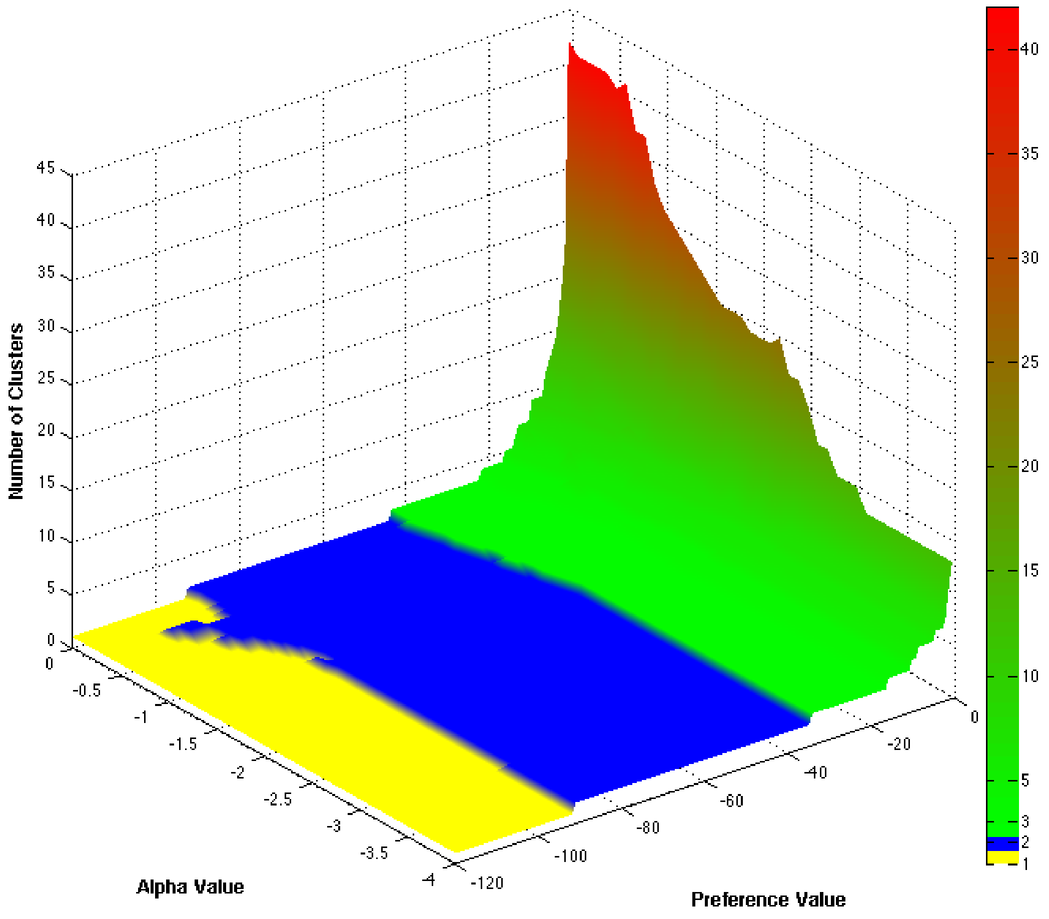


Figure 3-3: The number of states found by TSAP for each pair of parameter values is represented by the height of the surface. Two regions of this plot are of particular note. The parameter values for which TSAP finds one state are denoted by yellow. The parameter values for which TSAP finds two states are denoted by blue.

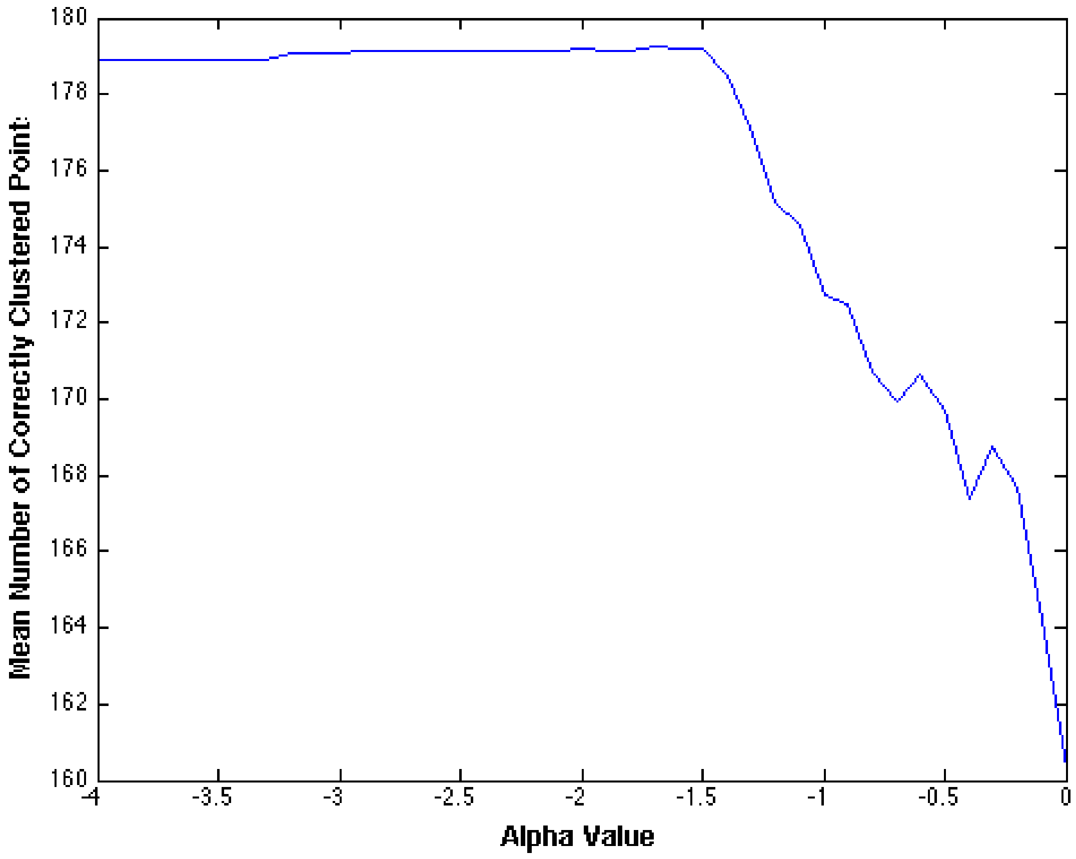


Figure 3-4: The number of correctly clustered points is greater for more negative alpha values.

3.1.2 Biologically Motivated Synthetic Data

To demonstrate the ability of TSAP to recover biologically meaningful results from genomic data, synthetic data were constructed to reflect patterns that are observed empirically in real data. Since running TSAP with an alpha value of 0 has been shown to be effectively the same as AP, only TSAP was applied to this synthetic data set and to the real data in the next section. Two types of fictional histone PTMs were designed to each take on a "high", "medium", or "low" state as shown in Fig. 3-5. Synthetic gene expression data were also constructed. Genes take one of five trajectories through the state space of histone PTM and expression patterns (Table 3.1). The synthetic data include observations at six time points for each gene.

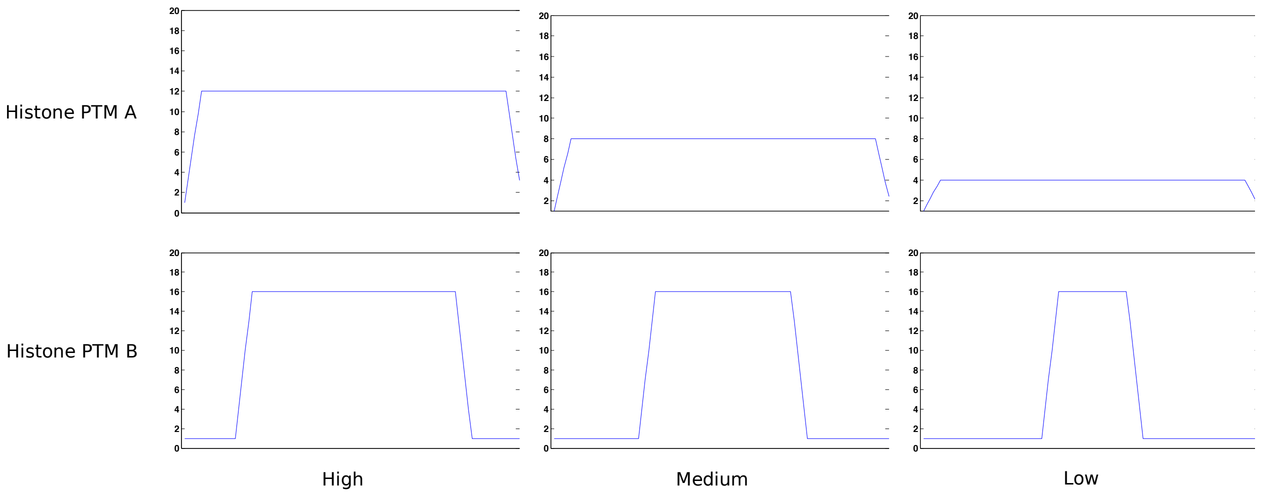
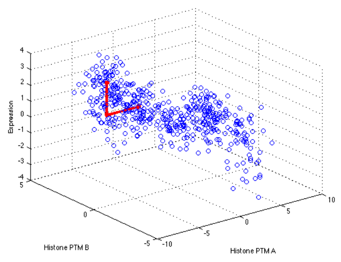


Figure 3-5: Synthetic histone PTM states.

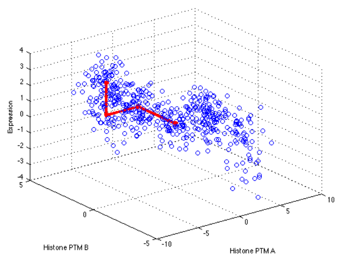
Feature vectors were calculated from the synthetic histone PTM and expression data as described in sections 2.4.1 and 2.4.2. To simulate the noise that would be present in real data, values sampled from a Gaussian distribution with mean 0 and variance $\frac{9}{16}$ were added to the components of the feature vectors. This amount of noise was enough to create overlap between the states. The paths taken by the five trajectories through feature space is shown in Fig. 3-6. The dimensions of the feature vectors were standardized to put them on the same scale. A similarity measure was then computed from the feature vectors as described in section 2.4.3.

1.	HML → HHL → HHL → MHL → MHM → MHM
2.	HMM → HMM → HMM → HHM → MHM → MHH
3.	HLO → HLO → HMO → HMO → MMO → MML
4.	LLO → LLO → MLO → MLO → MLO → MLO
5.	LMM → MMM → MMM → MLM → MLM → MLL

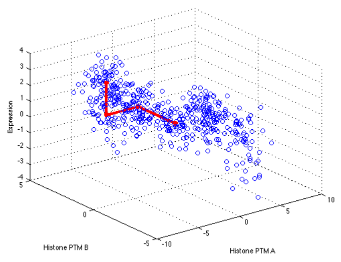
Table 3.1: The five trajectories contained in the synthetic data are represented in this table. Arrows separate each time point. The data at each time point are represented by three letters corresponding to the two histone PTMs and gene expression. H, M, L, and O stand for high, medium, low, and off, respectively.



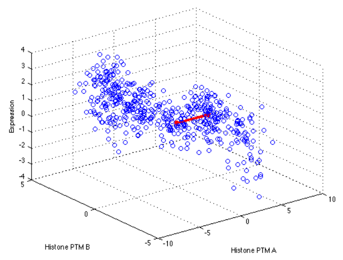
(a) Trajectory 1



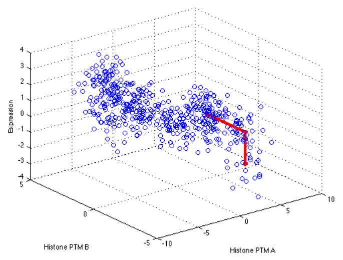
(b) Trajectory 2



(c) Trajectory 3



(d) Trajectory 4



(e) Trajectory 5

Figure 3-6: (a)-(d) show the feature vectors calculated from the biologically motivated synthetic data with noise added. The paths taken by the five trajectories are also shown. The five trajectories each have five time points, but visit fewer locations in feature space because each of the trajectories visit at least one location in feature space for more than one time point.

TSAP was run over a range of alpha values with preference values for which seven states are found at each alpha value. There are seven states present in the synthetic data, so the performance of TSAP when this number of states is found is of interest. As a measure of clustering performance, the fraction of time points at which each gene was correctly clustered was calculated from the results of each run of TSAP. This fraction is sometimes referred to as the *micro-averaged precision* [7] in the clustering literature. Each exemplar found by TSAP was associated with the state that the greatest number of observations assigned to that exemplar were designed to be part of in the synthetic data. Each gene at each time point was considered to be correctly clustered if it was assigned to the correct exemplar at that time point. For each alpha value, TSAP found seven exemplars for a range of preference values. Fig. 3-7 shows the median micro-averaged precision over a range of alpha values. The maximum median micro-averaged precision is achieved when the alpha value is -0.14. The median micro-averaged precision decreases for more negative alpha values because the assignment of genes to states is overly constrained. Thus, a moderate temporal constraint improves the performance of clustering on this biologically inspired synthetic data set.

3.2 Real Data

3.2.1 Description of Data Analyzed

The time series histone PTM and expression data analyzed in this section were generated through a collaborative effort between researchers at Columbia University and MIT. The cells used to provide material for the experiments were collected at specific stages of a directed differentiation protocol developed by Wichterle et al. [24] This protocol involves the subjection of embryoid bodies derived from mouse ES cells to specific inducers that simulate the neural identity specifying signals received by cells in vivo. During the course of this protocol cells were found to express markers of key stages of neural development. Furthermore, cells derived from ES cells with this protocol were shown to behave like embryonic motor neurons when grafted into live embryos. These results suggest that expression and histone PTM data

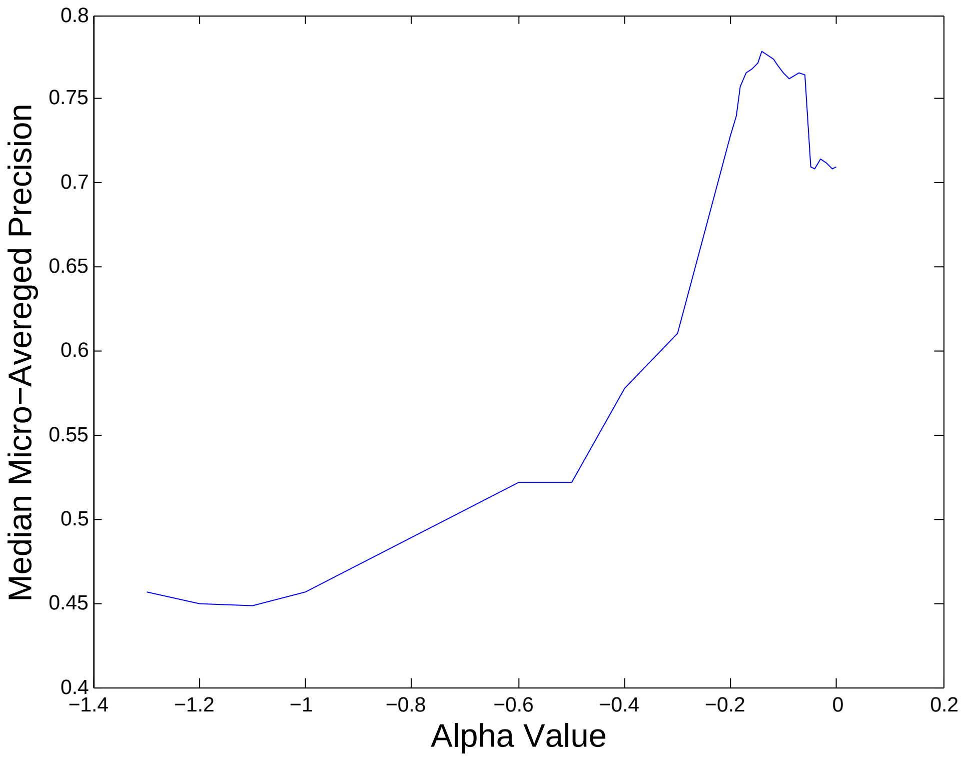


Figure 3-7: The median micro-averaged precision for a range of alpha values over preference values such that seven states were found.

measured in cells collected during the course of this directed differentiation protocol can be used to gain insight into the genomic activities involved in motor neuron development.

The time points included in the experiments were chosen relative to important steps in the protocol (Fig. 3-8). Cell samples were collected at each stage for analysis. The gene expression data were measured using Affymetrix microarray technology. Histone H3 lysine 4 trimethylation (H3K4me3) and histone H3 lysine 27 trimethylation (H3K27me3) were measured with a combination of chromatin immunoprecipitation (ChIP) using antibodies for the two marks and custom designed tiling microarrays made by Agilent. The Agilent arrays were designed to include genes from the Hox clusters as well as other genes that are expressed during motor neuron development. The reason for including the Hox genes in the array design will be described in section 3.2.3. A set of negative control genes were also included on the array.

A similarity measure over genes was constructed as described in Section 2.4. The resulting similarity measure was defined for 991 genes at 5 time points. The features extracted from the three sources of data are shown in Fig. 3-9. Table 3.2 shows that H3K27me3 is negatively correlated with both H3K4me3 and expression, while H3K4me3 is positively correlated with expression. This is in accordance with previous observations made about the relationship between these histone PTMs and expression.

Data	Correlation Coefficient
H3K27me3 vs. H3K4me3	-0.544
H3K27me3 vs. Expression	-0.3967
H3K4me3 vs. Expression	0.4952

Table 3.2: Correlation between data sources

3.2.2 Inferring States and Trajectories

TSAP was run over a range of parameter values using the constructed similarity measure. Fig. 3-10 shows the relationship between parameter values and the number of states that were found. Choosing a reasonable setting of parameters is a difficult task for most clus-

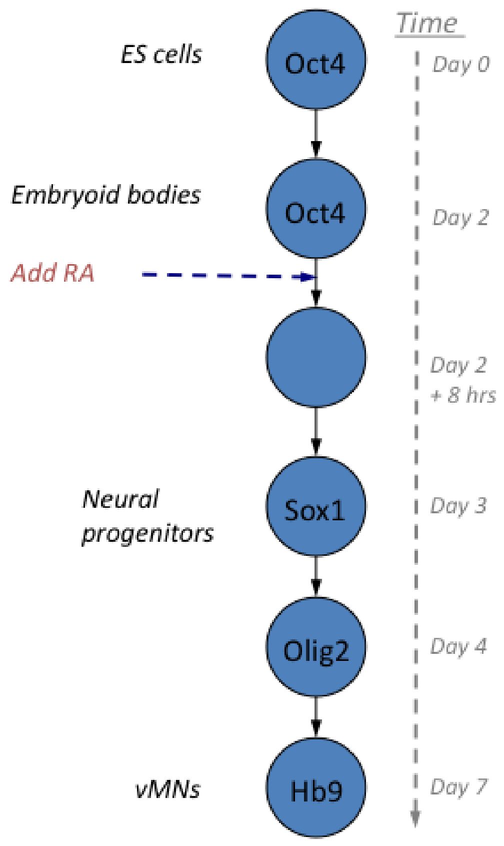


Figure 3-8: The time points included in the experiments are shown here. The gene names within the circles are genetic markers for stages of neural development that are expressed at the indicated time points. Days are numbered starting with untreated ES cells. After two days, the cells have formed embryoid bodies. Cells are collected immediately before and eight hours after the addition of retinoic acid (RA). By day 3 the cells are expressing Sox1 which is a marker for neural progenitors. By day 4, the cells are expressing Olig2 which is a marker for committed motor neuron progenitors. By day 7, the cells are expressing Hb9 which is a marker for postmitotic ventral motor neurons.

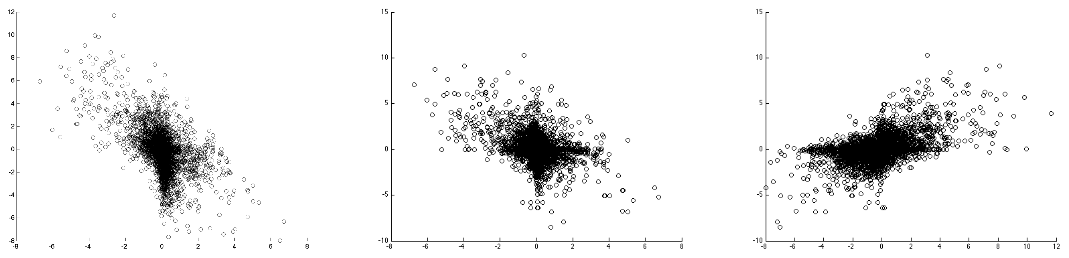

(a) H3K27me3 vs. H3K4me3 (b) H3K27me3 vs. Expression (c) H3K4me3 vs. Expression

Figure 3-9: Visualization of the features extracted from the data.

tering methods. It is reasonable to choose a preference value independently of the alpha value because the effect of the preference value on the number of states found is relatively independent of the alpha value. It can be seen in Fig. 3-10 that for a large range of the more negative preference values, the number of states found by TSAP is fairly stable. This reflects the organization of the data in that increasing the number of exemplars only improves the optimized value of the cost function when the preference value has increased by a relatively large amount. When the number of clusters becomes less stable for less negative preference values, this is evidence of overfitting. Thus, a preference value of -30.0 was chosen because this is one of the least negative preference values for which the number of states found by TSAP is still stable.

The alpha value influences how dissimilar consecutive data points in a trajectory must be in order for a state transition to be found by TSAP. A state transition will be made if the similarity of a data point to the exemplar of the new state plus the penalty for transitioning to the new state is still greater than the similarity of the data point to the exemplar of the old state. A state transition should not be made when the dissimilarity between consecutive data points in a trajectory is due to noise. If it is assumed that the top 10 percent of similarity values correspond to similarities between data points in the same state, the alpha value should be more negative than the top tenth percentile of the values taken by the similarity measure. This value is approximately -0.22. The alpha value should not be much more negative than this value or else clustering will be overly constrained. Thus, an alpha

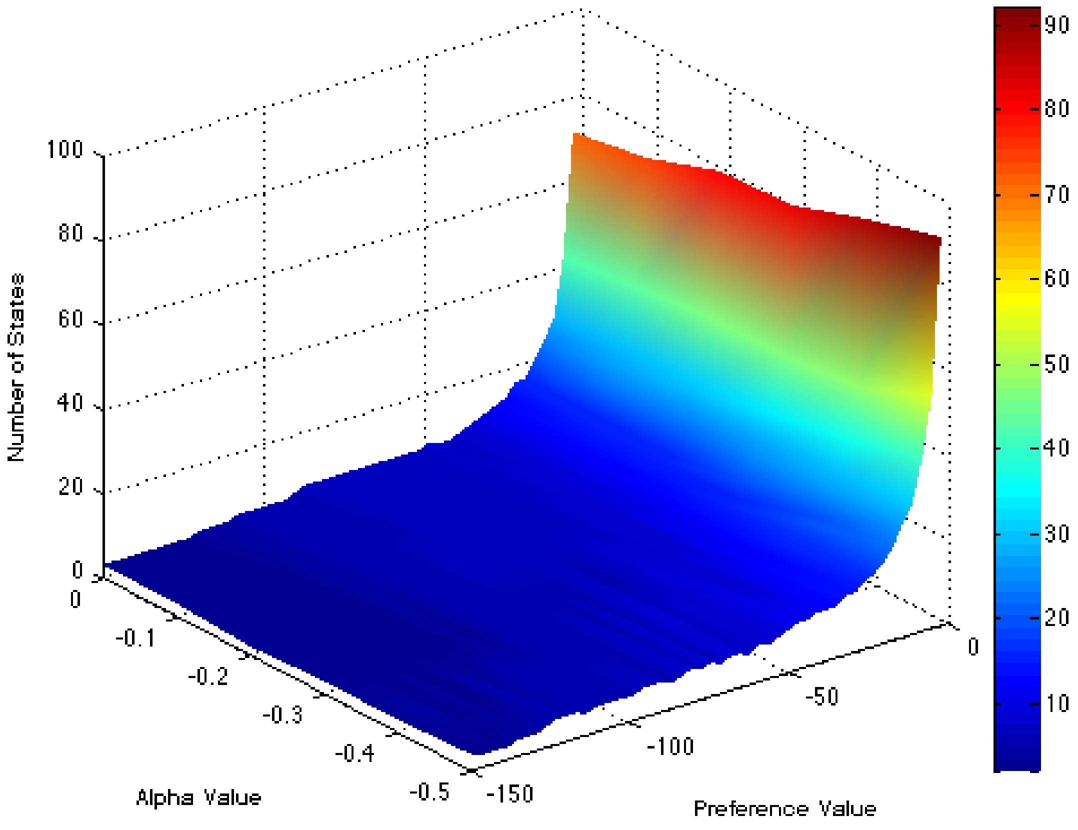


Figure 3-10: The alpha value varies between -0.5 and 0 and the preference value varies between -150 and 0. The number of states found by TSAP for each pair of parameter values is represented by the height of the surface. Note the large region of preference values for which the number of states found by TSAP is relatively stable.

value of -0.25 was chosen.

The exemplars found by TSAP are shown in Fig. 3-11. The state diagram in Fig. 3-12 represents the results found by TSAP. 264 unique trajectories were found by TSAP. The trajectories taken by the most genes are listed in Table 3.3. The most common trajectory is the trajectory taken through state 1 at all time points. This state represents the least amount of change from the ES cell stage. Thus, 210 of the genes in the data set are found by TSAP to stay relatively unchanged from their ES cell state throughout the entire time series.

Trajectory	Size
$< 1, 1, 1, 1, 1 >$	210
$< 7, 7, 7, 7, 7 >$	104
$< 1, 2, 2, 2, 2 >$	36
$< 1, 1, 1, 1, 5 >$	30
$< 7, 7, 7, 7, 8 >$	26
$< 6, 6, 6, 6, 6 >$	24
$< 8, 8, 8, 8, 8 >$	18
$< 5, 5, 5, 5, 5 >$	17
$< 4, 4, 4, 4, 4 >$	16
$< 7, 2, 2, 2, 2 >$	13

Table 3.3: Trajectories taken by the greatest number of genes.

One of the more common trajectories is the one that stays in state seven until day 7 at which point it transitions to state 8. This corresponds to a moderate decrease in H3K4me3 at all stages relative to the ES cell stage and then a fairly large decrease in expression at the last time point. Eight of the twenty-six genes that take this trajectory are associated with the gene ontology (GO) category for cell cycle. This is interesting because cells at day 7 of the differentiation protocol express Hb9 which is a genetic marker of post-mitotic ventral motor neurons. Thus, this trajectory captures the down-regulation of cell cycle related genes as cells leave the cell cycle. It is also interesting that the change in expression occurs significantly later than the change in H3K4me3.

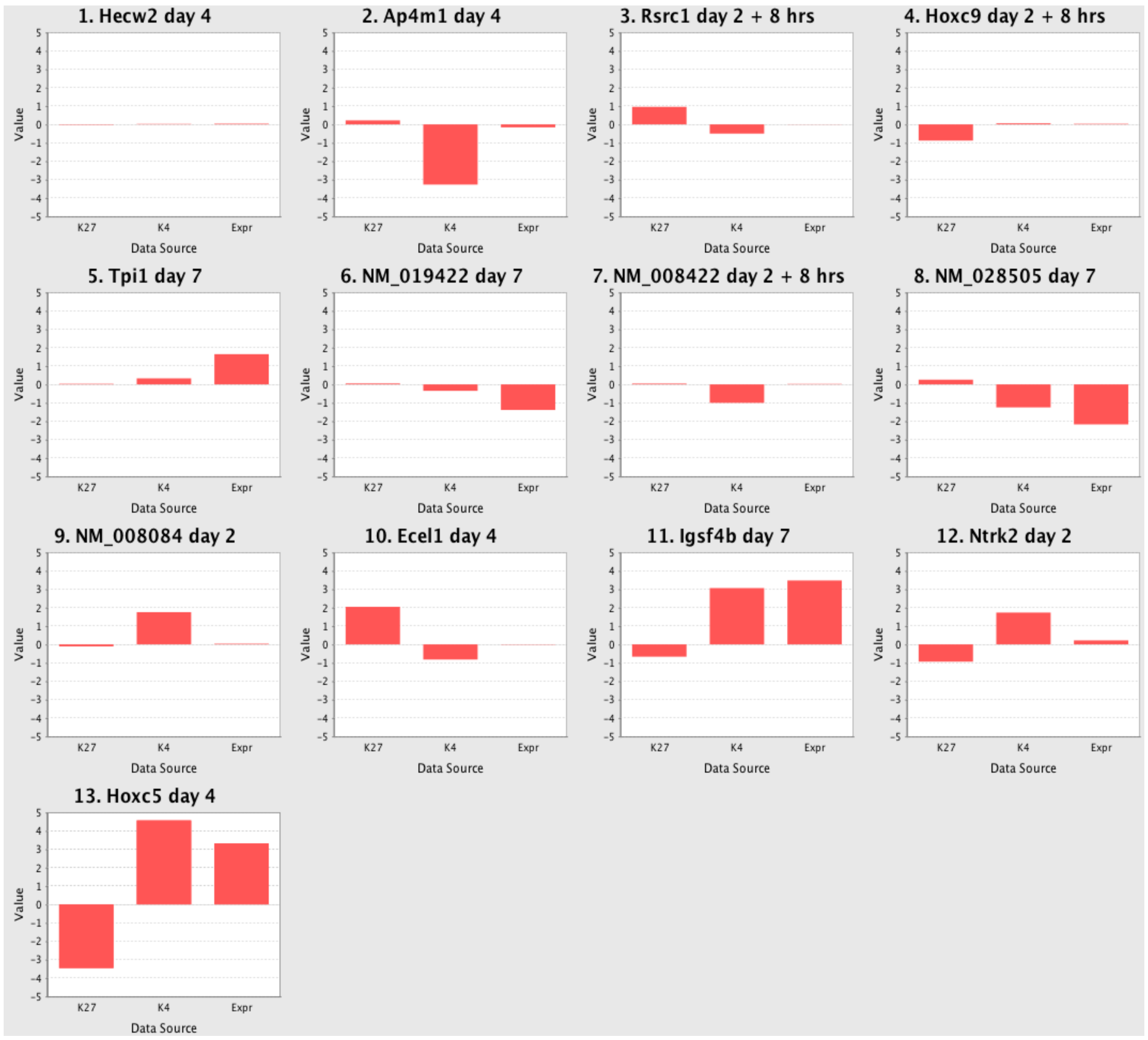


Figure 3-11: The exemplars found by TSAP. The title of each chart is the name of the gene and the time point at which it is being used as an exemplar. For each exemplar, the H3K27me3, H3K4me3, and expression features that were calculated for the similarity measure are shown as bars. Recall that the features represent an amount of change from the initial time point (ES cell stage). The exemplars can be taken as representative of the data points assigned to them by TSAP.

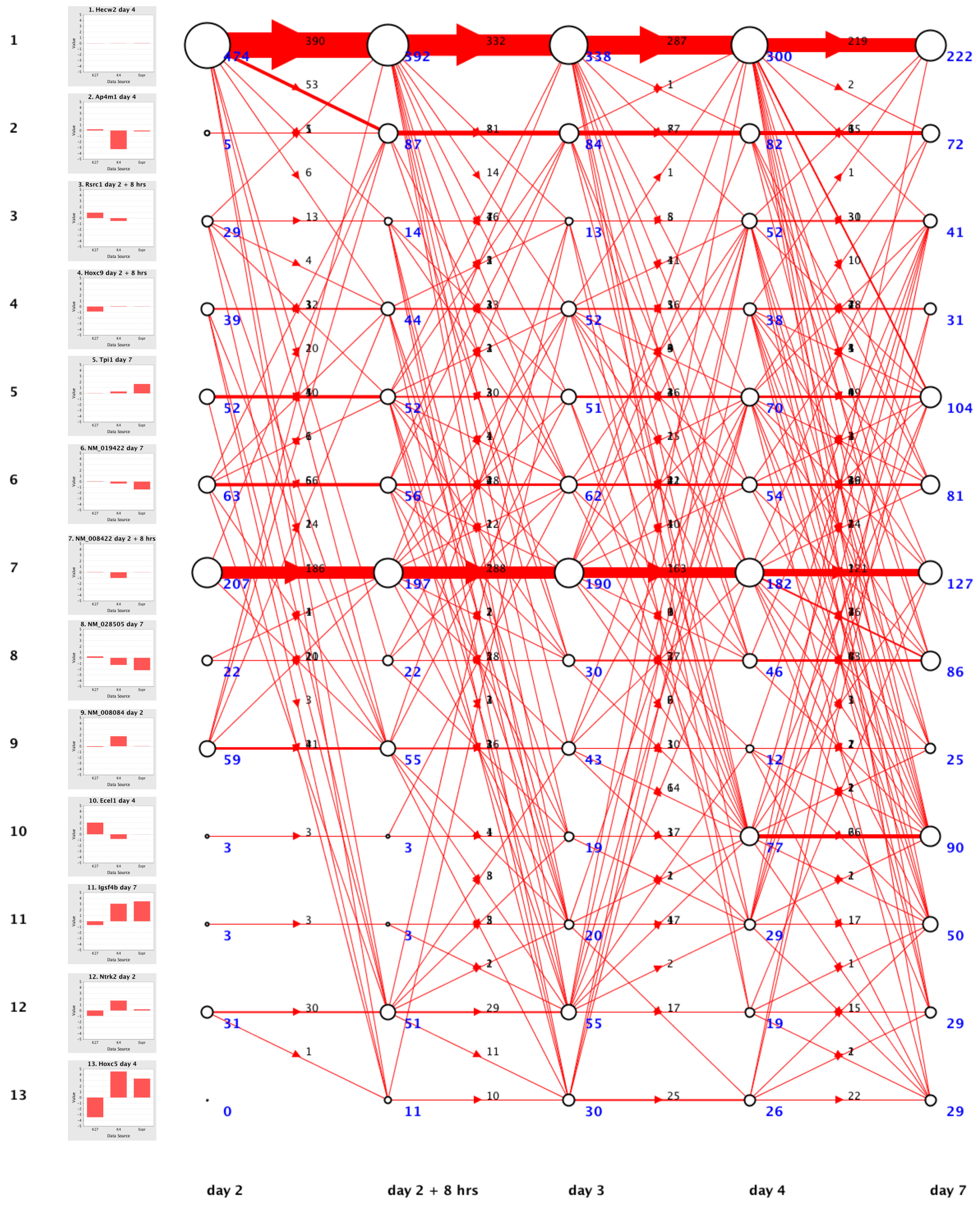


Figure 3-12: State diagram representing TSAP results for $\alpha = -0.25$ $pref = -30.0$

3.2.3 Regulatory Trajectories of the Hox Genes

A set of homeodomain transcription factor encoding genes known as the Hox genes are of particular importance to motor neuron development. There are four "clusters" of Hox genes in the mouse genome that are labeled Hoxa through Hoxd. The genes of each cluster are located within close proximity of each other in the genome, however the four clusters are each located on different chromosomes. Each Hox cluster contains genes numbered 1 through 13 with some missing from each cluster. Hox genes from the lower numbered end of each cluster (e.g. Hoxa1, Hoxc4, etc.) will be referred to as anterior whereas Hox genes from the opposite end of each cluster (e.g. Hoxb9, Hoxd13, etc.) will be referred to as posterior.

Within the organization of the central nervous system, cells are identified as belonging to specific neural subtypes. Subtype specification is important in the spinal cord where it has been observed that different motor neuron subtypes innervate different muscle targets. To a large extent, the identity of a motor neuron is reflected in the set of transcription factors expressed by the neuron's progenitors and by the neuron itself. In particular, spinal motor neurons exhibit patterns of Hox gene expression according to their position along the rostro-caudal axis of the spinal cord. Thus, motor neuron subtype identities are specified in part by the selective activities of the proteins encoded by the Hox genes [6]. The mechanisms by which the distinct patterns of Hox gene expression are established in motor neuron subtypes are not well understood. Thus, it is of interest to investigate the regulatory trajectories of the Hox genes during motor neuron development. A state diagram that has been filtered to include only the Hox genes is shown in Fig. 3-13.

The results found by TSAP suggest a potential regulatory mechanism involved in establishing the subtype specific Hox expression pattern. A majority of the more anterior Hox genes end up in a state by day 7 that is characterized by increased expression along with decreased levels of H3K27me3 and increased levels of H3K4me3. On the other hand, most of the more posterior Hox genes exhibit expression levels that have not changed seven days after the ES cell stage and are in a state that is characterized by increased H3K27me3 and decreased H3K4me3. Overall, we observe 24 different trajectories taken by the 33 Hox genes

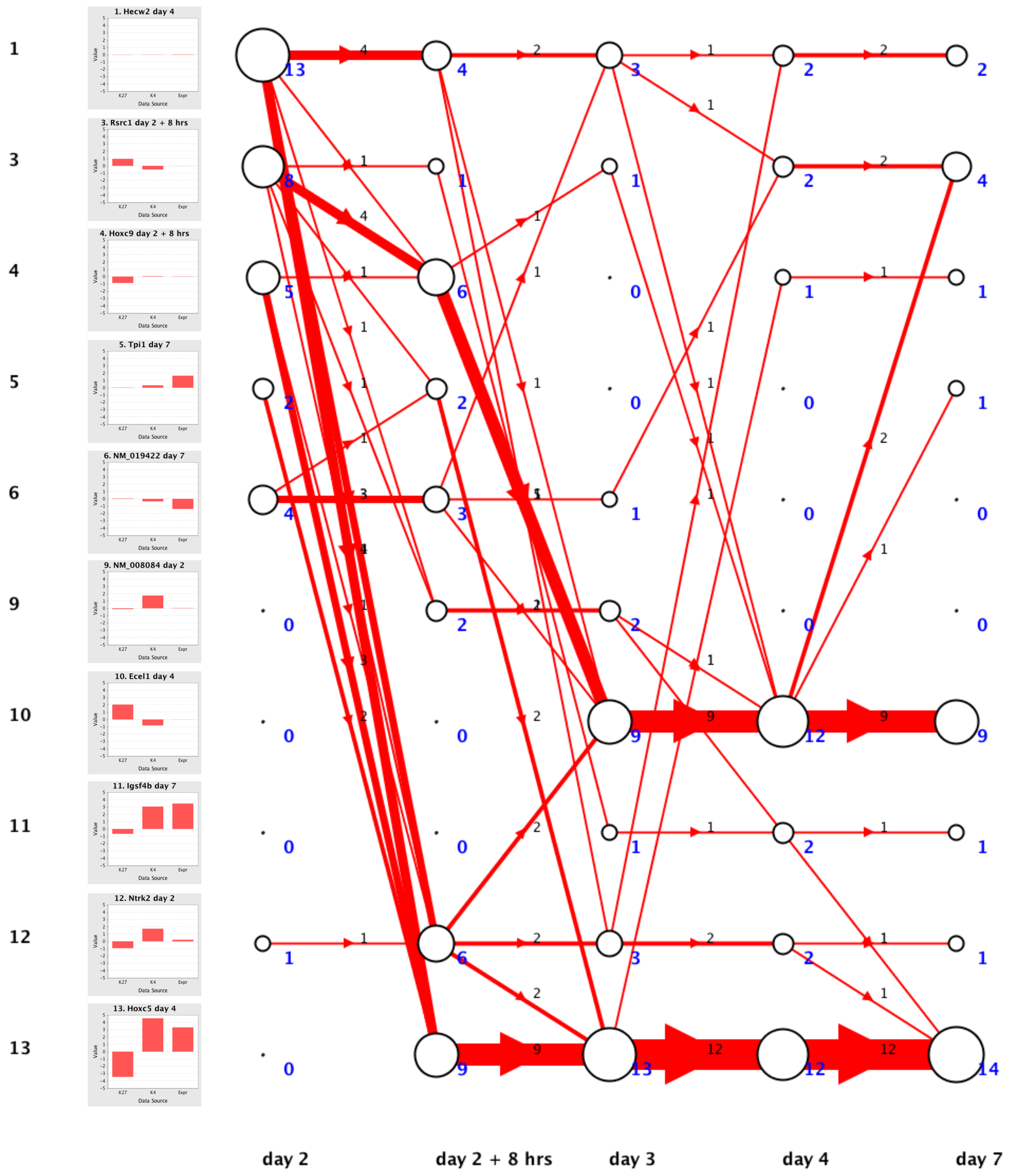


Figure 3-13: State diagram for genes from the Hox clusters

through 10 of the 13 states that are found by TSAP when run on the entire data set. However, 23 or more than half of the Hox genes end up in two of the thirteen states. One of the states, state 10, is characterized by an increase in H3K27me3 and a decrease in H3K4me3 with little change in expression as compared to the ES cell stage. We will refer to this state as the repressed state. The Hox genes that are assigned to this state by TSAP are all posterior Hox genes. The other state to which more than half of the Hox genes are assigned at the last time point is state 13. This state is characterized by decreased H3K27me3, increased H3K4me3, and increased expression. We will refer to this as the activated state. The Hox genes that are assigned to this state are all anterior to the Hox genes that were assigned to the repressed state at this time point. The difference between these two states reflects the expression pattern of anterior Hox genes that indicate the motor neuron subtype identity of the cells that are at the seventh day of the differentiation protocol.

State diagrams that are filtered to only include the Hox genes that are assigned to the repressed or activated states at the last time point are shown in Figs. 3-14 and 3-15 respectively. The trajectories in Fig. 3-14 demonstrate that most of the posterior Hox genes are assigned to the repressed state by day 3 and all of them are in the state by day 4. Days 3 and 4 express genetic markers for neural progenitors and committed progenitors respectively. The trajectories in Fig. 3-15 demonstrate that most of the anterior Hox genes have been assigned to the activated state by day 3. In fact, many of the anterior Hox genes are already in this state by 8 hours after the addition of RA. Interestingly, many of the trajectories of the posterior Hox genes pass through state 4 at the 8 hours after day 2 time point. This indicates that many of the posterior Hox genes exhibit decreased H3K27me3 after the addition of RA before gaining H3K27me3 and losing H3K4me3 by day 3. Overall, the trajectories of the anterior and posterior Hox genes indicate that most of the gene regulation involved in establishing the Hox gene expression patterns that reflect neural subtype identity has occured by the time cells are starting to express genetic markers of neural progenitors at day 3.

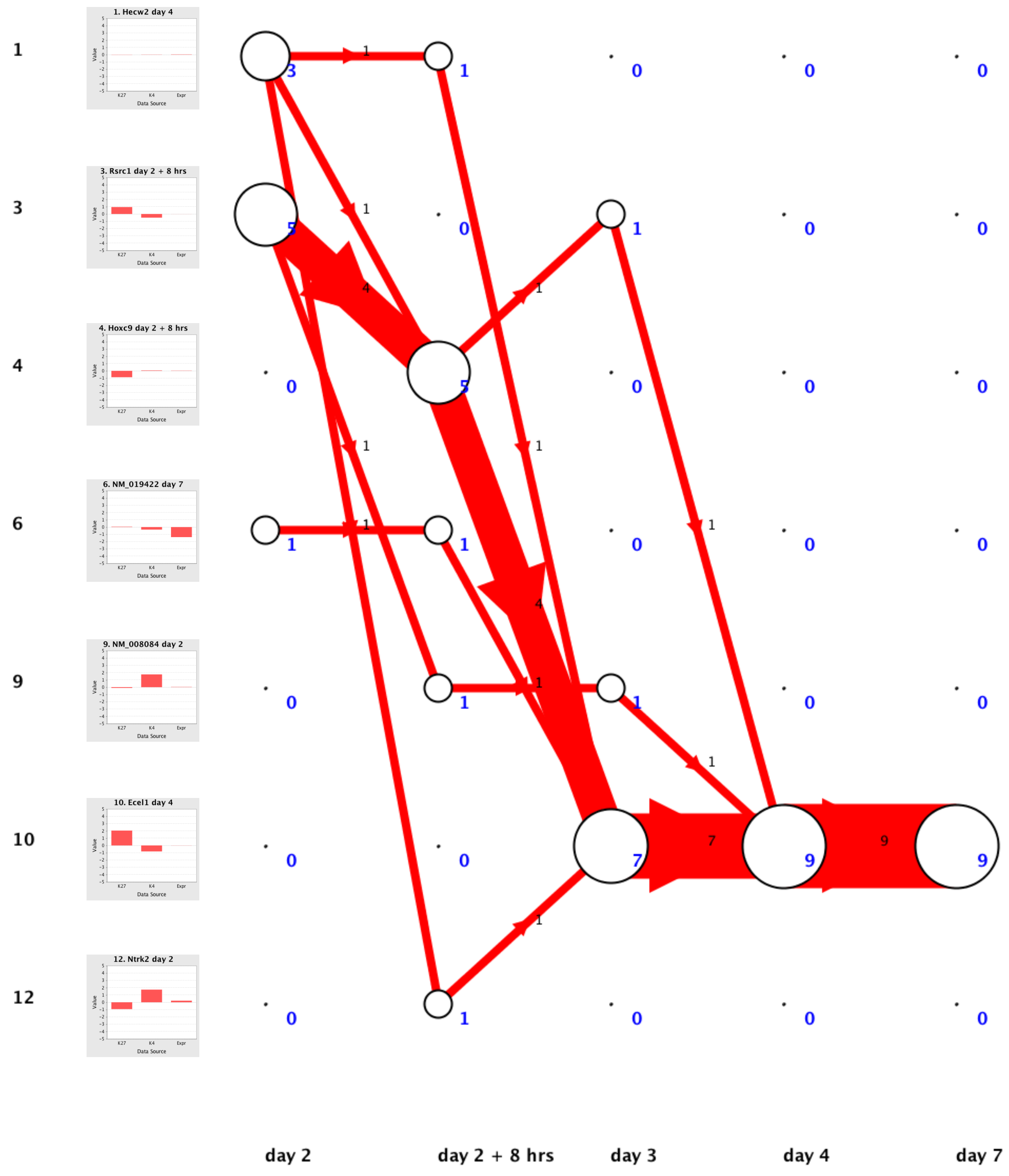


Figure 3-14: State diagram for Hox genes that are assigned to state 10 at day 7. These are all posterior Hox genes.

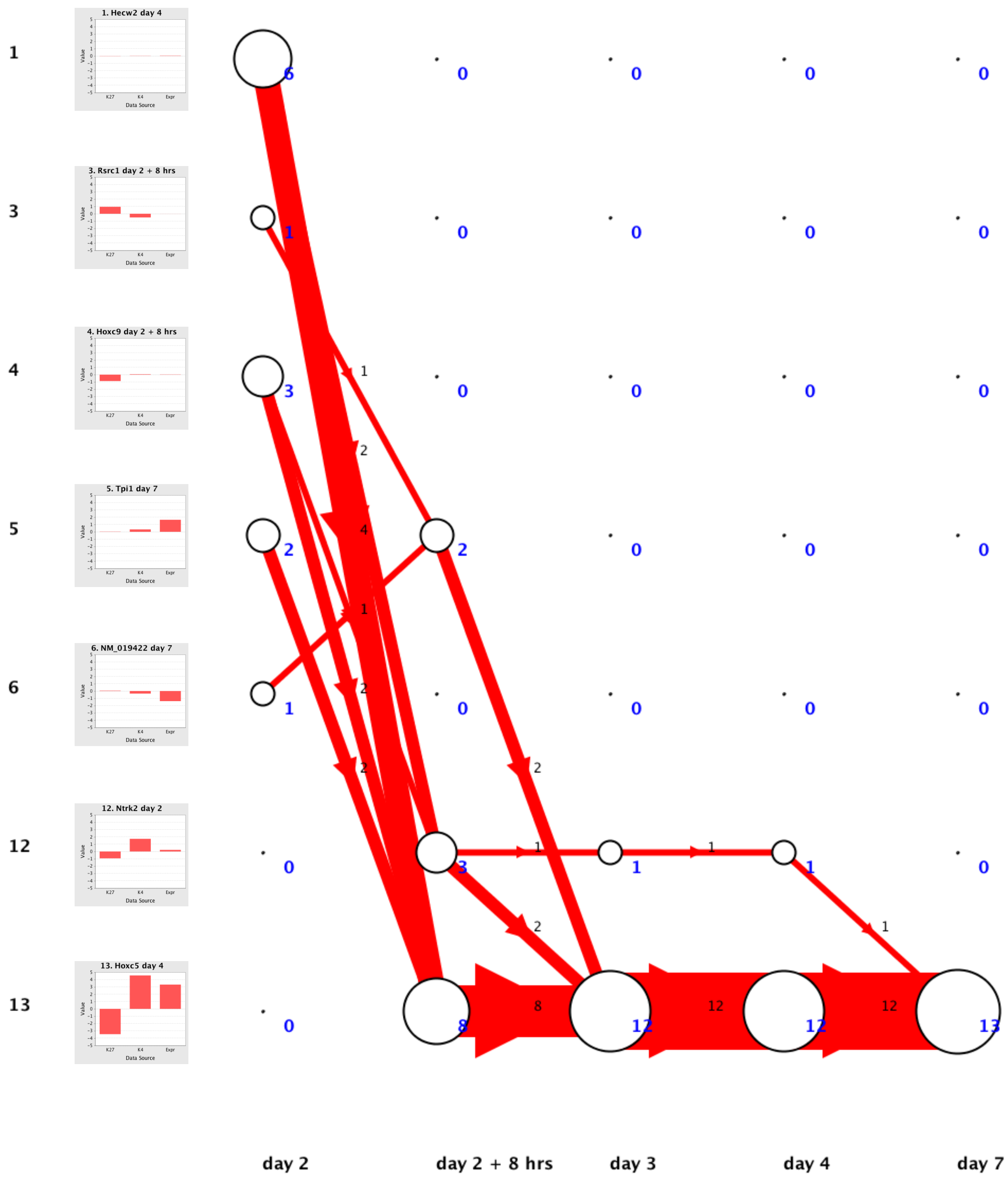


Figure 3-15: State diagram for Hox genes that are assigned to state 13 at day 7. These are all anterior Hox genes.

Chapter 4

Conclusion

In this thesis, we presented an approach to inferring regulatory states and trajectories from dynamic genomic data along with a method for constructing a similarity measure from gene expression and histone PTM data. We presented applications of TSAP to three data sets. We demonstrated with two synthetic data sets that TSAP is able to recover expected states and trajectories in a robust manner. Furthermore, by incorporating information about the temporal relationship among data points, TSAP was shown to outperform AP. We presented a third application of TSAP to a data set that reflected important stages of motor neuron development. By examining the trajectories taken by the Hox genes through regulatory states as found by TSAP, we were able to gain insight into the regulatory mechanisms that underly neural subtype specification.

4.1 Future Work

As the ability to manipulate embryonic stem cells improves and the use of technologies such as high-throughput sequencing [19] increases, the amount of dynamic genomic data will continue to grow. Thus, the ability to analyze temporal data using methods such as TSAP will become crucial to the task of uncovering the mechanisms behind gene regulation. Several improvements could be made to TSAP. One very useful improvement would be to incorporate a method for learning the α_1 and α_2 parameters of the $\gamma(\cdot,\cdot)$ function from the data.

A further improvement to the modeling approach taken by TSAP could be made based on the observation that the current $\gamma(\cdot,\cdot)$ function could be generalized. In the current proposed method, a distinction is only drawn between transitions that stay in the same state and transitions between different states. Conceivably, if parameters for a generalized $\gamma(\cdot,\cdot)$ function could be learned, the function could draw a distinction between each possible different transition between states. This would make the manner in which TSAP models transitions between states even more similar to a HMM while still maintaining the advantages that TSAP holds over HMMs mentioned in section 1.4. The problem of inferring states and trajectories from dynamic data arises in other fields besides gene regulation. Thus, it will be of interest to apply TSAP to additional data sets.

Bibliography

[1] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden markov model. In *NIPS*, volume 14, 2002.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings 1993 IEEE International Conference on Communications*, 1993.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, New York, first edition, 2006.

[4] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19:185–193, 2003.

[5] M. Charikar, S. Guha, É. Tardos, and D. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65:129–149, Aug 2002.

[6] J. S. Dasen, B. C. Tice, S. Brenner-Morton, and T. M. Jessell. A hox regulatory network establishes motor neuron pool identity and target-muscle connectivity. *Cell*, 123:477–491, Nov 2005.

[7] I. S. Dhillon, S. Mallela, and D. S. Modha. Information theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD*, pages 89–98. ACM Press, 2003.

[8] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95:14863–14868, Dec 1998.

[9] J. Ernst, O. Vainas, C. T. Harbison, I. Simon, and Z. Bar-Joseph. Reconstructing dynamic regulatory maps. *Molecular Systems Biology*, 3:74, 2007.

[10] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, Feb 2007.

[11] S. F. Gilbert. *Developmental Biology.* Sinauer, Sunderland, Massachusetts, eighth edition, 2006.

[12] G. Grant, E. Manduchi, and C. Stoeckert. *Current Protocols in Molecular Biology*, chapter 19. John Wiley and Sons, Inc, Jan 2007.

[13] J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21:3201–3212, Aug 2005.

[14] D. J. Huebert, M. Kamal, A. ODonovana, and B. E. Bernstein. Genome-wide analysis of histone modifications by chip-on-chip. *Methods*, 40:365–369, Dec 2006.

[15] M. I. Jordan. Graphical models. *Statistical Science*, 19:140–155, 2004.

[16] B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden markov models. *AT&T Technical Journal*, 64, Feb 1985.

[17] T. Kouzarides. Chromatin modifications and their function. *Cell*, 128:693–705, Feb 2007.

[18] F. Kschischang, B. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47, Feb 2001.

[19] T. Mikkelsen, M. Ku, D. Jaffe, B. Issac, E. Lieberman, G. Giannoukos, P. Alvarez, W. Brockman, T. Kim, R. Koche, W. Lee, E. Mendenhall, A. O'Donovan, A. Presser, C. Russ, X. Xie, A. Meissner, M. Wernig, R. Jaenisch, C. Nusbaum, E. Lander, and B. Bernstein. Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, advanced online publication:553–560, Jul 2007.

[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Francisco, California, revised second edition, 1988.

[21] B. Ren, F. Robert, J. Wyrick, O. Aparicio, E. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannett, E. Kanin, T. Volkert, C. Wilson, S. Bell, and R. Young. Genome-wide location and function of dna binding proteins. *Science*, 290:2306–2309, Dec 2000.

[22] T. Richmond and C. Davey. The structure of dna in the nucleosome core. *Nature*, 423:145–150, May 2003.

[23] M. Schena, D. Shalon, R. Davis, and P. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270:467–470, Oct 1995.

[24] H. Wichterle, I. Lieberam, J. Porter, and T. Jessell. Directed differentiation of embryonic stem cells into motor neurons. *Cell*, 110:385–397, Aug 2002.

[25] Z. Wu, R.A. Irizarry, R. Gentleman, F.M. Murillo, and F. Spencer. A model based background adjustment for oligonucleotide expression arrays. Technical Report TR-2001-22, John Hopkins University, Department of Biostatistics, Baltimore, MD, 2003.

[26] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi electric research laboratories, Cambridge, Massachusetts, Jan 2002.