The University of Sheffield

Transfer Report

A Gaussian Process model for Transcription Factor Activity of *Caenorhabditis elegans*

Author:
Muhammad A. Rahman

Supervisor:
Neil D. Lawrence

A report submitted in fulfilment of the requirements for the Transfer to Doctor of Philosophy

in the

Machine Learning Research Group
Department of Computer Science

December 2014

THE UNIVERSITY OF SHEFFIELD

Abstract

Faculty of Engineering
Department of Computer Science

Transfer to Doctor of Philosophy

A Gaussian Process model for Transcription Factor Activity of *Caenorhabditis elegans*

by Muhammad A. Rahman

In molecular biology and genetics, a transcription factor is a protein that binds to specific DNA sequences and controls the flow of genetic information from DNA to mRNA. To develop models of cellular processes, quantitative estimation of the regulatory relationship between transcription factors and genes is a basic requirement. But quantitative estimation is complex due to some reasons. Many of the transcription factors' activity and their own transcription level are post transcriptionally modified; very often the levels of the transcription factors' expressions are low and also contain noise. So, from the expression levels of their target genes it is useful to infer the activity of the transcription factors. Here we develop a Gaussian process based regression to infer the exact TFAs from a combination of mRNA expression level and DNA protein binding measurement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 System Biology

The prime goal of Biology is to get the insight of various principle and detail of biological systems. More than six decade ago, Watson and Crick discover the structure of DNA (Watson and Crick [1953]) and radically changed the way of study and development of biology and biological systems. They explained the biological phenomena with the help of molecular basis. This new concept help to explain different aspect of biology like heredity, different disease, various evolutionary aspects as well as development with more firm theoretical ground. Since then, biology became a framework of knowledge governed by some basic and fundamental laws of physics.

Due to the enormous advancement of molecular biology, at present we have in-depth knowledge of elementary processes like evolution, heredity, disease, development etc. These mechanisms also includes other biological features like replication, transcription, translation, and so on. Accomplishment of symbolic DNA sequencing helped to reveal large number of genes and their transcriptional products. DNA sequences for many organisms like *Mycoplasma, Plasmodium falciparum, Saccharomyces cerevisiae, Caenorhabditis elegans, Drosophila melanogaster, Homo sapiens* and many more have been fully identified. Due to the advancement of different methods gene expression profile are available at the mRNA level. Even measurement of protein level and their different subsequent actions are also making progress.

Undoubtedly understanding at the molecular level will accelerate to understand the biological systems but these knowledge isn't sufficient to understand biological systems as systems. Genes and protein are few components of a whole system. It is necessary to understand what constitute the system, but even only this knowledge is not sufficient

to understand the complete system. System biology is a new field of biology to acquire understanding up to system level of biological system (Kitano [2000]).

The extent of the area of system biology is very broad and various technique may be may be required for each individual research target. Very often it demands combined effort from multiple discipline research area like molecular biology, high-precision measurement technology, mathematics, computer science, control theory and other engineering and scientific field. Kitano [2002] mentioned the main four key areas to carried out the research: (1) genomics and other molecular biology research, (2) various technology for comprehensive and high-precision measurements, (3) computational studies, such as bioinformatics, modelling and simulation, software tools, and (4) analysis of the dynamics of the systems. This clearly depicts requirement of multi-disciplinary research effort to get the knowledge of biological systems as systems. The abstract concept of system yet more than a collection of multi-disciplinary research components. To obtain the proper insight of system beside the detail description of the components it is also essential to know what happens during the period or processes when any stimuli and/or disruptions take place.

Identification of the system structure is the primary requirement to understand biological system. Some of the key structure might be different regulatory relationship of genes and interactions with protein that shows the metabolism pathway and signal transduction, physical structure of chromatin, cells, organisms and other components. Though it is very critical to monitor biological processes in bulk using high-throughput DNA micro-array, real-time polymerase chain reaction (RT-PCR), protein chips and other methods thereafter methods to identify genes and metabolism network have to be established. Once a system structure is established up to a certain degree, it is required to find out the behaviour. To understand the behaviour properly a number of analysis method can be used. For example, if we want to know the sensitivity of a specified behaviour against some external perturbations, and its time to return its normal state since the stimuli take place. This type of analysis provides the system level characteristics as well as uncover important insights of medical treatments by revealing cell response to certain chemical affinities.

To apply the knowledge obtained from system structure and behaviour understanding, further research is required to control the state of the biological systems. All these phases leads toward the establishment of technologies those allow us to design biological system which can provide cures for different diseases. Even some futuristic example could be organ cloning technique for the treatment of diseases what require organ transplants or building biological materials for engineering specially robotics having self-sustaining and self-repairing capabilities.

1.2 Dynamic Mathematical model: what and why in System biology?

Any models are abstractions of reality. Models mostly designed to focus of specific aspects of the objects for certain kind of study. usually other aspects are abstracted away. Biologists almost regularly making use of tangible 'real world' models. Some of them are very simple like molecular ball-and-stick, again some of them are complex such as animal disease model or model organisms. They also use 'conceptual models'. These conceptual models usually take the form of verbal descriptions of the system and communicate by diagrams. These diagrams are usually constructed with a set of components and the ways they interact with each other. While representing knowledge of cellular or different other processes, these interaction diagrams, or 'cartoon' models may play a central role (Ingalls [2012]).

A major drawback of these cartoon models is that while considering system behaviour they could be significantly ambiguous. It even more, if there is any interaction network related with feedback. Complexity increases even further when the number of components and their corresponding interactions in the network grows. Sometimes it become very difficult to get the intuitive understanding of the system behaviour. But a mathematical model or description of the same model can eliminate uncertainty of the model behaviour. The mathematical model will consider the quantitative representation of individual interaction of the cartoon model. In Figure 1.1 species A and B bind to form a new complex. The newly formed complex hinder the rate at which molecules of species C are transformed to species D. A numerical description of the process is required to quantify the interaction. Though for simple cases only equilibrium condition

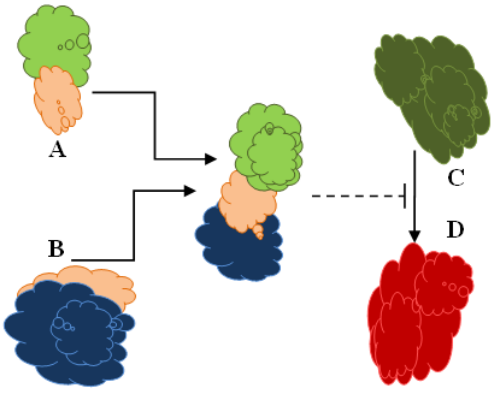


FIGURE 1.1: A 'cartoon' model of protein protein interaction. Two different molecular species A and B bind to form a complex molecular. The newly formed complex hinder the rate at which molecules of species C are transformed to species D.

is enough, but in many other cases binding and unbinding rates might be also required. The cartoon model or traditional knowledge cannot provide a quantitative description rather a qualitative explanation of the molecular interaction. But a well studied mechanisms with sufficient data might be capable to show the quantitative characteristics. The interaction diagram with related quantitative data can be used to develop a dynamic mathematical model. This kind of model consists a number of equations that describe the systems behaviour over time. This behaviour is termed as "system's dynamic behaviour". These models are usually *mechanistic*, as they explain the mechanisms of molecular interaction with some laws of physics and chemistry as well as mathematics. Any of the part of the mechanistic model actually represent the real observed system. Any change in the mechanistic model's component will also mimic to the real system. So, model simulation (*in silico* experiments) can be used to predict system behaviour. Some numerical software built with different programming language are used for this simulation purposes.

As mathematical model is a hypothesis, so the outcome or result of the model hypothesis are also hypothesis. Though the real cellular behaviour definitively cannot predict by simulation, but they can be invaluable for further experimental design by showing the promising paths for further investigation, or by showing the inconsistencies between the real laboratory observations and our understanding about the model or system.

1.3 The Systeome Project

"Systeome" is an collection of system profile for all genetic variations and environmental stimuli response. A system profile consists of a set of information about the properties of the system including structure, behaviour, analysis of result such as bifurcation diagram or phase portfolio. The structure of the system should include structure of gene and metabolic networks and its physical structure, associated constants, and their properties (Kitano [2002]).

Systeome is not just a simple cascade map rather it assumes different active and dynamic solutions, simulations as well as profiling of various system status. The Systeome project might be established with dealing all aspects for profiling the Systeome of yeast, *C. elegans, Drosophila,* mouse and finally human. The primary goal of the Human Systeome project is defined as- "To complete a detailed and comprehensive simulation model of the human cell at an estimated error margin of 20 percent by the year 2020, and to finish identifying the system profile for all genetic variations, drug responses, and environmental stimuli by the year 2030"(Kitano [2002]).

This is a highly ambitious project, and requires several milestones. Some pilot projects will lead toward the final goal. Initial pilot projects are using yeast for the simplicity of structure and subsequent behaviour. *C.elegans* have comparatively complex system structure and so is their behaviour. Beside such pilot projects concurrently the Human Systeome project shall be commenced.

The futuristic impact of this project will be very wide spread as well as far-reaching. These will be the baseline and standard asset for any further biological research to provide fundamental diagnostics and prediction for a variety of medical practice. This Systeome project involves many other major engineering projects for developing the measurements, as well as software platform.

1.4 *Caenorhabditis elegans*

Caenorhabditis elegans is a nonparasitic, soil dwelling, small nematode worm. *C. elegans* and other *Caenorhabditis* species are found through all over the world. It can easily colonize mostly in the rotting materials with other microorganism. *C. elegans* is easy to maintain in the petri dishes at the laboratory. At 25 °C *C. elegans* completeits life cycle in just 2.5 days from fertilized embryos to egg-laying adult through 4 larval stages. Its typical life span is 2-3 weeks. In 1965, Sydney Brenner introduced *Caenorhabditis elegans* as a model organism to study the behaviour and development of animal (Brenner [1974]).

C. elegans is a relatively new addition as a model organism but its biological characteristics and property already been studied to an extraordinary level. The anatomical characteristics and detail development of this nematode was facilitated by its simple body plan. Its an eukaryote and it shares cellular and molecular structures and control pathways with higher organism. *C. elegans* is multicellular, an adult wild type consist of 959 somatic cells and among these 302 are neurons (Palikaras and Tavernarakis [2013], Sulston and Horvitz [1977]). Its developmental process like embryogenesis, morphogenesis goes through a complex process to growth to an adult. Yet monitoring of the cellular process and recording of cell division pattern is comparatively easier as its body is transparent. C Elegan's complete cell lineage at the electron microscopy level has been completed. Its already been established and this cell lineage is remarkably invariant between animal to animal (Brenner [1974], Byerly et al. [1976], Sulston et al. [1980], Wood [1988]).

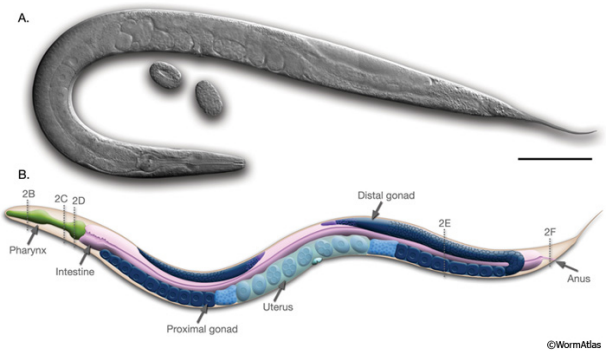


FIGURE 1.2: Anatomy of an adult hermaphrodite(*C.elegans*). A. DIC image of an adult hermaphrodite, left lateral side. Scale bar 0.1 mm. B. Schematic drawing of anatomical structures, left lateral side (Courtesy of WormAtlas http://www.wormatlas.org/hermaphrodite/introduction/IMAGES/introfig1leg.htm).

To elucidate pathways and processes relevant to human biology and disease C. Elegans is using as a vital model. There are between $\sim 20,250$ to $\sim 21,700$ predicted protein-coding genes in *C. elegans* (Gerstein et al. [2010]). Using four different orthology-prediction methods Shaye and Greenwald [2011] assayed four methods to compile a list of *C. elegans* orthologs of human genes. A list of 7,663 unique protein-coding genes were resulted in that list and this represents $\sim 38\%$ of the 20,250 protein-coding genes predicted in *C. elegans*. When human genes introduced into *C. elegans* human genes replaced their homologs. On the contrary, many *C. elegans* genes can function with great deal of similarity to human like mammalian genes. So, the biological insight acquire from *C. elegans* may be directly applicable to more complex organism like human.

1.5 Transcription

All the process in the body take places through some proteins. So, cells needs protein continuously. On the consequences, protein are required to be manufactured at every moment in the body. Inside cell protein is manufactured from the DNA. When any cells in the body is in need of protein, a special signal is sent to the DNA using hormones. Then proteins reside in DNA start to manufacture based on the received codes. The way that the enzymes finds the information required for protein construction is extremely complicated.

DNA (Deoxyribonucleic acid) transcription is a process that involves the transcribing of genetic information from DNA to a complementary RNA (Ribonucleic acid). Protein is produced from the copy of DNA by the transcription process. This production of proteins and enzymes are controlled by the coding of cellular activity. Even the conversion of

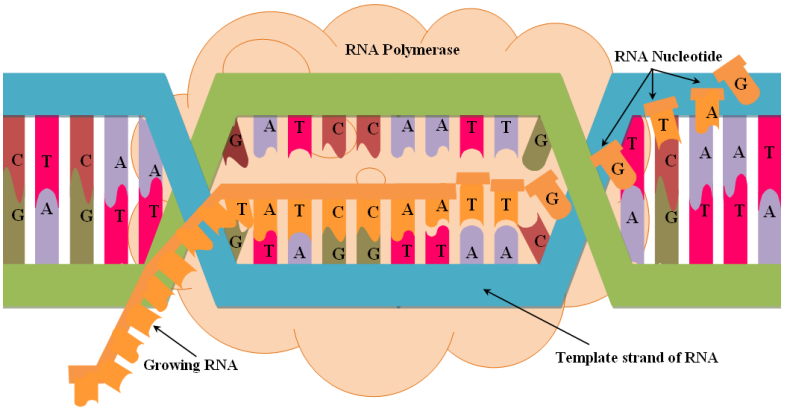


FIGURE 1.3: DNA Transcription

DNA to proteins is not straight forward. An RNA polymerase read the sequence of DNA, which produce an complementary RNA. DNA consists of four nucleotide bases named adenine (A), guanine (G), cytosine (C) and thymine (T) that are paired together (A-T and C-G) to give DNA its double helical shape. The major steps to the process of DNA transcription are :

RNA polymerase binds to DNA: In order to initiate the DNA transcription RNA polymerase and sigma factor form a holoenzyme. Transcription process starts at the promoter region of a double-stranded DNA. Sigma factor can recognize the DNA and its promoter region.

Elongation: A sequence specific DNA binding factors called transcription factors then unwind the DNA strand. Elongation of the transcript then continues by the RNA polymerase and a sequence of chain is opened up. A messenger RNA (mRNA) is formed when RNA polymerase transcribe into a single stranded RNA polymer from a single strand of DNA.

Termination: RNA polymerase moves along the DNA unwinding its double helical form until it reaches the terminator sequence. At that point, RNA polymerase detaches from the DNA and releases the mRNA polymer. In this way DNA double helix is opened, transcribed and reclosed with minimum stress on the DNA molecule. At any certain time many RNA polymerase can transcribe a single DNA sequence, which can manufacture a large quantity of protein at once.

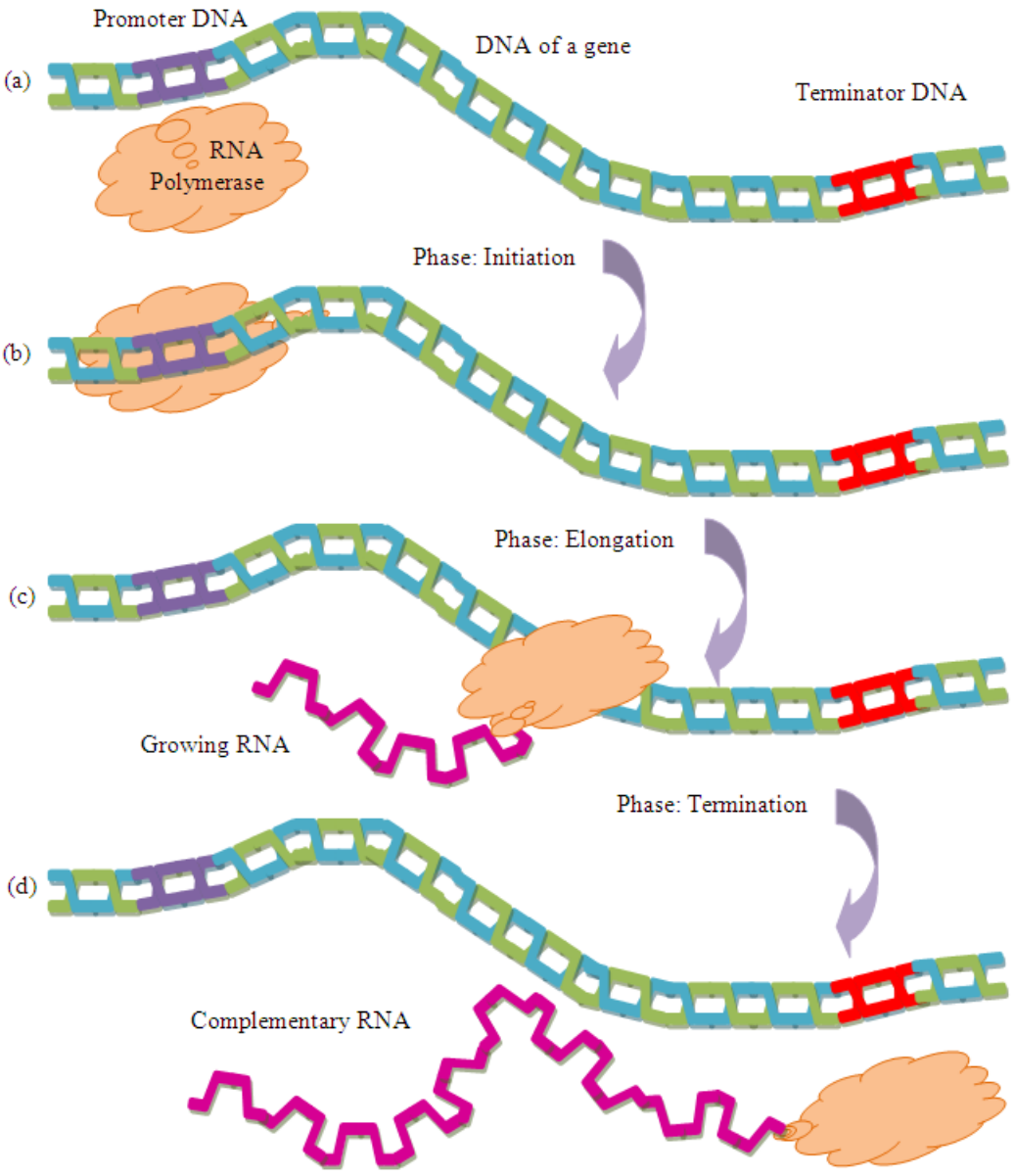


FIGURE 1.4: Transcription Process: DNA Transcribed in mRNA

1.6 Transcription Factor

A transcription factor is a protein that binds to DNA sequences and controls the flow of genetic information coding from DNA to mRNA (Karin [1990], Latchman [1997]). Transcription factors can both promote or block the transcription process and act as an activator or repressor respectively (Lee and Young [2000], Nikolov and Burley [1997], Roeder [1996]). A transcription factors may contain one or more DNA-binding domains. These binding domains attach to specific sequences of DNA adjacent to the genes that they regulate. Though some other protein such as coactivators, deacetylases, chromatin

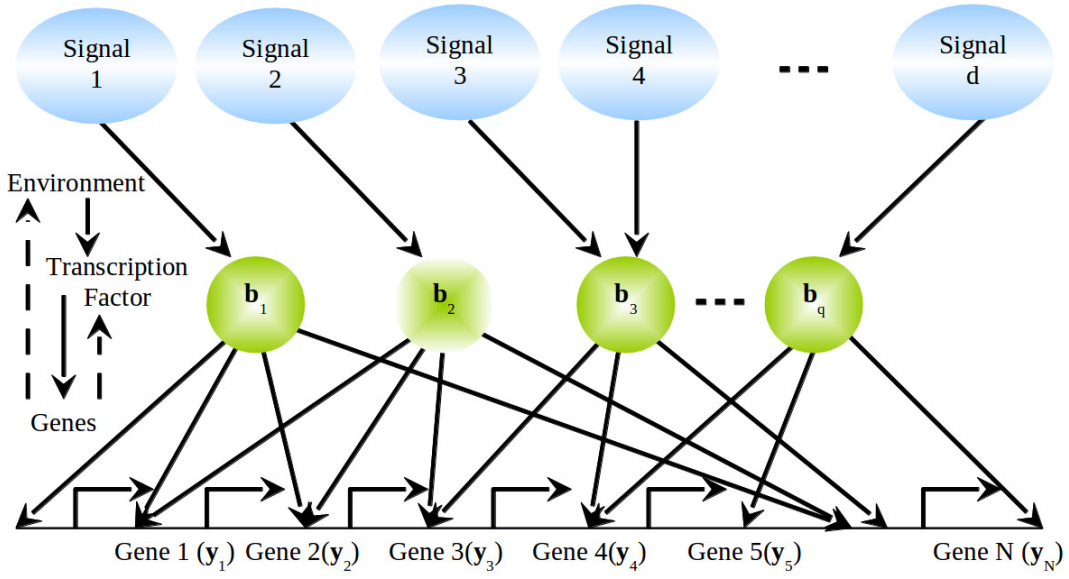


FIGURE 1.5: The mapping between environmental signal, transcription factor inside the cell the genes that they regulate.

remodelers, kinases, histone acetylases, and methylases also play crucial roles in gene regulation but due to lack of DNA-binding domains they are not classified as transcription factors (Brivanlou and Darnell [2002], Mitchell and Tjian [1989], Ptashne and Gann [1997]). Figure 1.5 describes the mapping (we can also say "cartoon" mapping) between the environmental signal, transcription factors inside the cell, and the gene that they regulate. The environmental signal activates specific transcription factor proteins. After the activation the transcription factors bind DNA to change the transcription rate (the rate at which mRNA is produced) of specific target genes. The mRNA is then translated into protein by the process named translation (Alon [2006]).

1.7 Key Research Questions

Key Research questions:

- Can we develop a robust data driven dynamic system for gene specific transcription factor activities using Gaussian process over a mechanistic model?
- Can we step forward to find out the transcription factor activities from a unicellular microorganism to multicellular eukaryote?
- Is it possible to build some replicate specific robust clusters of the gene expression data for *C. elegans*?

Chapter 2

Dynamic Modelling of TFAs

In modern molecular biology the biological systems like cells are treated as a complex systems. The usual conception of the complex system is a very large number of simple but identical elements interact to generate the complex behaviour. But the actual behaviour of biological systems are different from this conception. A vast number of functionally different and multifunctional group of elements act with each other selectively, perhaps nonlinearly to generate coherent instead of complex behaviour. Mostly, functions of biological systems depend on a combination of the network and specific elements involved.

Development of molecular biology has discovered a large number of biological facts like sequencing genome, protein properties etc. But to explain the biological systems behaviour only these are not sufficient. Study of cell tissues, organs, organisms etc. are also the systems of components to consider and their specific interaction which is defined by the evolution could be more supportive to reach the prime goal of biology. Though advancement in more accurate quantitative experimental approach will continue, but the detail functional insights of biological systems may not give the exact results from purely intuitive basis due to the intrinsic complexity of biological systems. A proper combination of experimental and computational approaches is more likely to solve this problem.

In modern molecular biology the organisational and functional activity of gene regulatory network is a key experimental and computational challenge.

2.1 Gene Expression Data for Genomics

Living cells contains thousands of genes. These genes codes for one or more protein. Expression of these genes are regulated by many of these protein through a very complex regulatory pathway. Usually this regulation occurred to accommodate the change of the environment, as well as through the cell cycle of the development process. Gene expression is the process where information contains in the gene is used to synthesis a functional gene product. The genetic code stored in the DNA usually expressed or interpreted by gene expression which represents the phenotype. These gene expression data are usually stored in DNA microarray or DNA chip which is also known as biochip.

Figure 2.1 shows two Affymetrix chips which contains DNA microarray. A match is shown at the bottom for the purpose of size reference of a microarray. The solid-phase DNA macroarray is usually a collection of ordered microscopic spots called features. Figure 2.2 shows the schematic of the gene expression microarray data. On a typical Affymetrix microarray there are 6.5 million locations (represented by columns) with millions of identical DNA strands in every locations. Every strand construct with 25 probes or bases. The microarray is rinsed and washed with fluorescent stain. To accomplish a DNA test, two types of samples are used one is controlled sample and another is test sample. After extracting mRNA from DNA, copied are made from mRNA by reverse transcription. Two different fluorescent tagged with cyanide are usually used to differentiate between controlled samples and test samples. In general, green is used for controlled copy and red for test copy. Then the tagged samples are washed on the microarray. DNA is analysed based on matching with the probes on the microarray. A laser is used to glow the fluorescent molecules. After the hybridisation process a green spot represents a hybridisation with the controlled targets only, a red spot indicates hybridisation with the test targets only, yellow represent hybridisation both with the

FIGURE 2.1: Gene Expression Data: Affymetrix Micro Array (Image courtesy Wikipedia. http://en.wikipedia.org/wiki/DNA_microarray)

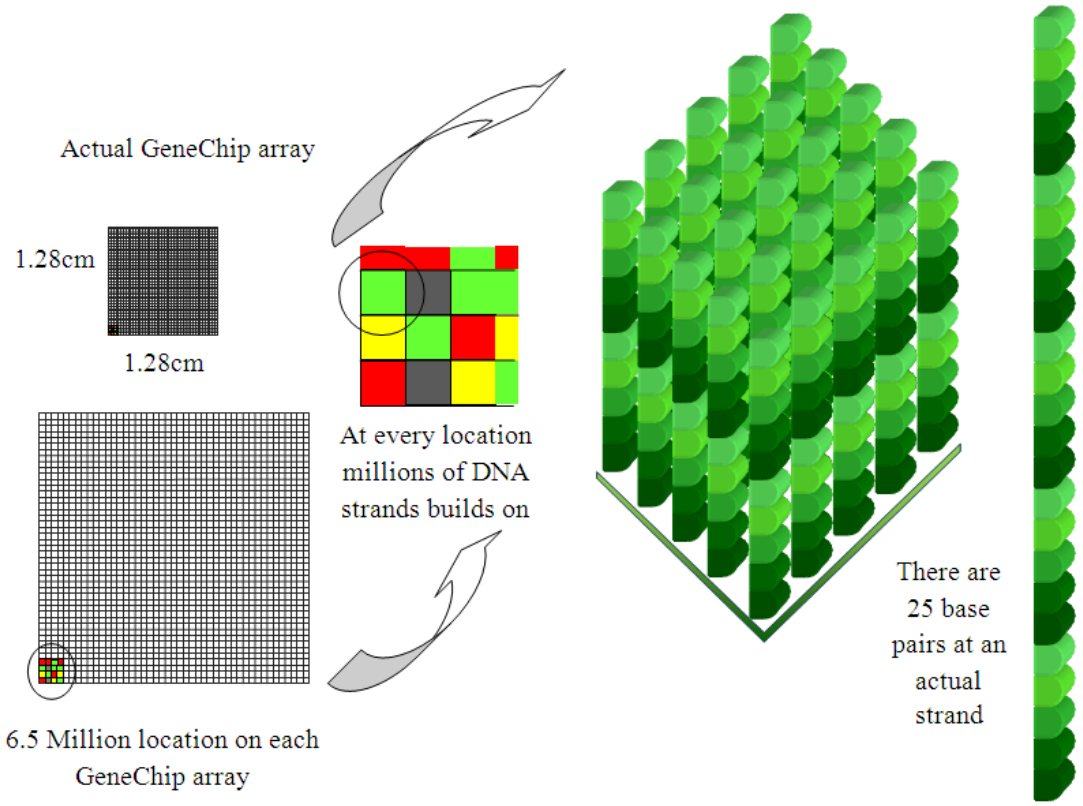


FIGURE 2.2: Gene Expression Data: Affymetrix Micro Array

controlled targets and test targets, and black represents hybridisation with the neither samples, i.e. no hybridisation. Over the last couple of decades these gene expression data became one of the key resource of the biologists to diagnose diseases and drug discovery, gene discovery and determining genetic variations, aligning and comparing genetic codes, biomerker development, forensic application, functional analysis and also in the field of computational biology.

Ong et al. [2002] modelled the regulatory pathway in E.coli from the time series gene expression microarray data by modelling causality, feedback loops or hidden variables using a Dynamic Bayesian network and tried to gain the insight of regulatory pathway. By analysing gene expression data Friedman et al. [2000] were the first to determine the properties of transcriptional program for Baker's yeast using a Bayesian network.

Many of the recent studies already established the fact that the gene function of regulatory network depends on qualitative as well as quantitative aspects of the organisation of the network like high throughput data, including genomic sequence, expression profiles and transcription factor.

Among them one of the major challenges is to quantitative measurement and analysis of the mechanisms regulating mRNA transcription. Though using high throughput techniques it is comparatively easier to measure the output of transcription, but it is experimentally very complicated to measure the protein concentration levels of transcription factors and chemical affinity to the genes. Very often transcription factors are post-transcriptionally modified. So, the actual protein concentration levels and binding affinities could be an unreliable proxy of the mRNA expression levels of transcription factors (Sanguinetti et al. [2006]).

Due to the advancement of the experimental technique lot of interest in recent years has been growing to infer information about regulatory activity from target genes. Now biologists can acquire the information about the structure of the transcriptional regulatory network. Lee et al. [2002] determined the transcriptional regulatory network of yeast using chromatin immunoprecipitation(ChIP). They tried to figure out how yeast transcriptional regulators bind to promoter sequences across the genome. By calculating a confidence value (P value) and setting up specific threshold they considered the protein-DNA interactions and artificially imposes a binding or not binding binary decision for each of the protein- DNA pair.

2.2 Connectivity Information

Xie et al. [2005] used motif conservation information for higher organisms like human, dog, rat and mouse. For promoter analysis they considered a number of network motif (also known as transcription factor binding sites) and also some new motifs. These type of data termed as connectivity data Liao et al. [2003] provide information about whether a certain transcription factor can bind the promoter region of a gene or not.

2.3 Different models of TFAs

In recent years most methods aim to infer a matrix of transcription factor activities (TFAs). These TFAs are sum up in a single number at a certain experimental point to find the concentration of the transcription factor and its binding affinity to its target genes. Many of the researcher used different ways or algorithm to find out these TFAs. For example, Liao et al. [2003] developed a data decomposition technique with dimension reduction and introduced 'network component analysis'. This method takes account of the connectivity information by imposing algebraic constraints on the factors. They argued that classical statistical methods such as principal component analysis and independent component analysis dose not consider the underlying network structure while

computing low dimensional or hidden representation of a high-dimensional data sets like DNA microarray.

Alter and Golub [2004] used a dimension reduction technique (SVD) to figure out TFAs and also the correlation between DNA replication initiation and RNA transcription during the yeast cell cycle. Using multivariate regression and backward variable selection to identify active transcription factors Gao et al. [2004] targeted the same; Boulesteix and Strimmer [2005] used partial least squares (PLS) regression to infer the true TFAs from a combination of tRNA expression and DNA protein binding measurement. A major drawback of the above mentioned methods is that transcription factor activities do not hold any information regarding the strength of the regulators interactivity between the transcription factor and its different target genes. But it is expected that depending on the experimental conditions transcription factor activities can vary from gene to gene. Even it is also expected that different transcription factors may bind the same gene. In most of the cases, realistic information about the intervals may not be true as they were not based on fully probabilistic model. Moreover, false positives are always a problem for connectivity data, typically a large portion of Chip data suffers form it (Boulesteix and Strimmer [2005]). Furthermore, due to the various cellular process or changes in environmental conditions the structure of the regulatory network of the cell can change considerably. Using regression-based methods it is difficult to track these changes. Nachman et al. [2004] build a probabilistic model, using the basic framework of dynamic Bayesian networks using discrete random variables for protein concentrations and binding affinities. Though the model was more realistic but the computational complexity for genome-wide analysis can be expensive.

2.4 Our Goal

We will propose a dynamic model that extends the linear regression model of Liao et al. [2003] and probabilistic model of Sanguinetti et al. [2006] to model the distribution of each transcription factor acting on each gene. We will model the temporal changes in the gene-specific TFAs from time-series gene expression data using Gaussian process (a stochastic process whose consciousness comes from random values and where the random variables has a normal distribution and it is associated with every single point in a range of times or of space; *Chapter 4* contains detail explanation). The covariance structure of the transcription factors will be shared among all genes. This approach will lead to a manageable parameter space and figure out useful information about the correlation of TFAs.

Initially to build our model we will use two datasets: the classical yeast cell cycle dataset of Spellman et al. [1998] and the yeast metabolic cycle dataset of Tu et al. [2005]. Both of the data sets were used to study the above mentioned models. So, these data will be a source of useful comparisons. In both cases the connectivity data will be Chip data (Lee et al. [2002]; Harbison et al. [2004]). Finally we will use the data set of Caenorhabditis elegans (*C. elegans*) to obtain a deeper insight. *C. elegans* was used to build the probabilistic functional gene network Network.

Chapter 3

Probabilistic Dynamic Modelling

3.1 Probabilistic Modelling

We have developed our *R* based tools *chipDyno* based on the probabilistic approach of Sanguinetti et al. [2006]. First we will give a brief introduction of that approach then we will present results.

The logged gene expression measurements are collected in a design matrix , $\mathbf{Y} \in \mathbb{R}^{\mathbf{N}\times\mathbf{d}}$ where N is the number of genes and d the number of experiments. The connectivity measurements are collected in a binary matrix $\mathbf{X} \in \mathbb{R}^{\mathbf{N}\times\mathbf{q}}$, where q is the number of transcription factors; element (i, j) of $\mathbf{X}$ is one if transcription factor j can bind gene i, zero otherwise.

In Sanguinetti et al. [2006], TFAs are obtained by regressing the gene expressions using the connectivity information, giving the following linear model-

$$\mathbf{y_n} = \mathbf{B_n}\mathbf{x_n} + \boldsymbol{\epsilon_n} \tag{3.1}$$

Here $n = 1, ..., N$ indexes the gene, $\mathbf{y_n} = \mathbf{Y}(n, :)^T$, $\mathbf{x_n} = \mathbf{X}(n, :)^T$ and $\boldsymbol{\epsilon_n}$ is an error term. The matrix $\mathbf{B_n}$ has d rows and q columns, and models the gene specific TFAs.

As different TFAs for every individual gene will increase number of model parameters drastically. But through marginalization by prior distribution on the rows of $\mathbf{B}_n$ these parameters can be dealt. Two plausible assumptions for selecting the prior distribution will be helpful to determine the gene specific TFAs. Firstly, $\mathbf{b}_{nt}$ has the Markov property and hence gene specific TFA $\mathbf{b}_{nt}$ at time t depends solely on the gene specific TFA at time $(t - 1)$ and the second assumption is, the prior distribution to be stationary in time.

To satisfy these conditions then there will be two limiting conditions of prior distributions- Assume all the $\mathbf{b}_{nt}$ are identical, then the first limiting case will take place. So that

$$\mathbf{b}_{n1} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.2}$$

and

$$\mathbf{b}_{n(t+1)} \sim \mathcal{N}(\mathbf{b}_{nt}, \mathbf{0}) \tag{3.3}$$

If the experimental dataset comes by replicating a condition then this model will be an appropriate model.The second limiting case appears when all the $\mathbf{b_{nt}}$ are independent and identically distributed-

$$\mathbf{b}_{nt} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.4}$$

This is the case when experimental dataset comes from independent samples drawn without any temporal order.

Sanguinetti et al. [2006] expected a realistic model of time series data to be somewhere in between this two extremes-

$$\mathbf{b}_{n(t+1)} \sim \mathcal{N}(\gamma \mathbf{b}_{nt} + (1-\gamma)\boldsymbol{\mu}, (1-\gamma^2)\boldsymbol{\Sigma}) \tag{3.5}$$

for $t = 1, ..., (d-1)$ and $\mathbf{b}_{n1} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ Where γ is a parameter measuring the degree of temporal continuity of the TFAs.

If genes are independent for given TFA then the likelihood function is given by:

$$p\left(\mathbf{Y}|\mathbf{B}, \mathbf{X}\right) = \prod_{n=1}^{N} p\left(\mathbf{y}_n|\mathbf{B}_n, \mathbf{x}_n\right) \tag{3.6}$$

TFAs can be estimated a posteriori using Bayes's Theorem-

$$p\left(\mathbf{b}_n|\mathbf{Y}\right) = \frac{p\left(\mathbf{Y}|\mathbf{b}_n\right) p\left(\mathbf{b}_n\right)}{p\left(\mathbf{Y}\right)} \tag{3.7}$$

3.2 Datasets

Sanguinetti et al. [2006] has done their experiments on yeast's cell cycle data of Spellman et al. [1998] which is a unicellular microorganism. One of our research key question was can we step forward to find out the transcription factor activities from a unicellular microorganism to multicellular eukaryote. *C. elegans* is a established multicellular eukaryotic model organism. To find out the TFA of *C. elegans* basically we had to

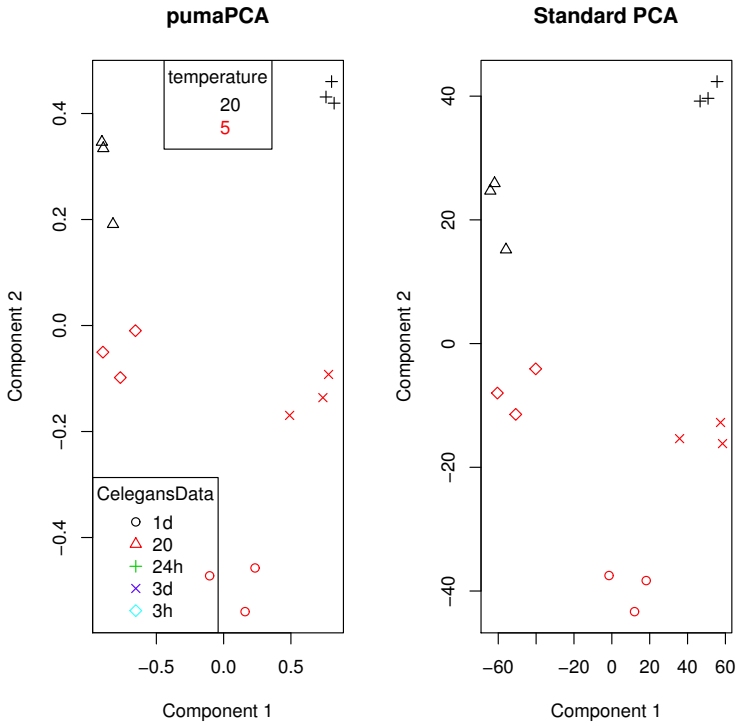


FIGURE 3.1: Principal component analysis of gene expression time series data

work with three type of datasets. *i*). Time series data *ii*). Transcription Factors *iii*). Connectivity information between genes and transcription factors.

3.2.1 Time series data

The gene expression data files came from the Prof. Andrew Cossins's lab, Institute of Integrative Biology, University of Liverpool 15 Affymetrix single colour GeneChip data on point estimate of expression level came without estimates of uncertainty level. To extract this data we use the *puma* package (Pearson et al. [2009]). The experimental data had 5 different time points. Apart from the temperature rest of the environmental condition was same for the consistent result. The experimental data was collected within one day of its adulthood at the temperature 20 °C. To measure the gene response to chill exposure then the temperature was reduced to 5 °C and experiments were done after one hour, then after 1 day (24 hour), then again after 3 days (72 hours) and for final experiments the temperature was set back to 20 °C and data were collected within one day of rise of temperature. All the experiments were repeated two more times. i.e. we have 3 independent replicates of similar experiments. Figure 3.1 shows the PCA analysis of the time series data.

3.2.2 Transcription Factors

From different data sources we found different number/list of transcription factor. Inmaculada et al. [2007] build a database named *C. elegans* differential gene expression database (EDGEdb) which contains the sequence information about 934 predicted transcription factors and their DNA binding domains. Initially we took these 934 transcription factors for our baseline experimental setup but we also kept the opening to deal with any number of transcription factor depending on the requirement/ update of the sequence information of transcription factors.

3.2.3 Connectivity Information

WormNet [2014] is a gene network of protein-encoding genes for *C. elegans* based on based on probabilistic function and modified Bayesian integration. They have considered 15,139 genes and 999,367 linkages between genes associated with a log-likelihood score (LLS). These measured scores represents a true functional linkage between a pair of genes Lee et al. [2007]. The linkage between two genes were measured based on the following evidence codes(WormNet [2014])-

- CE-CC Co-citation of worm gene
- CE-CX Co-expression among worm genes
- CE-GN Gene neighbourhoods of bacterial and archaeal orthologs of worm genes
- CE-GT Worm genetic interactions
- CE-LC Literature curated worm protein physical interactions
- CE-PG Co-inheritance of bacterial and archaeal orthologs of worm genes
- CE-YH High-throughput yeast 2-hybrid assays among worm genes
- DM-PI Fly protein physical interactions
- HS-CC Co-citation of human genes
- HS-CX Co-expression among human genes
- HS-DC Co-occurrence of domains among human proteins
- HS-LC Literature curated human protein physical interactions
- HS-MS human protein complexes from affinity purification/mass spectrometry

- HS-YH High-throughput yeast 2-hybrid assays among human genes
- SC-CC Co-citation of yeast genes
- SC-CX Co-expression among yeast genes
- SC-DC Co-occurrence of domains among yeast proteins
- SC-GT Yeast genetic interactions
- SC-LC Literature curated yeast protein physical interactions
- SC-MS Yeast protein complexes from affinity purification/mass spectrometry
- SC-TS Yeast protein interactions inferred from tertiary structures of complexes

We have constructed the connectivity matrix between genes and associated transcription factors from the gene to gene linkage and log-likelihood scores. We choose Co-expression among worm genes (CE-CX), High-throughput yeast 2-hybrid assays among worm genes (CE-YH), Literature curated human protein physical interactions (HS-LC) and High-throughput yeast 2-hybrid assays among human genes (HS-YH) to start our experiments. But if needed we can consider any of the evidence to reconstruct the connectivity matrix. From the gene list we have picked the protein-coding genes (i.e. transcription factors) and later binarized it. If there is an associated LLS value between a gene and a transcription factor we set the value '1' and '0' otherwise.

3.3 Result Analysis

We have developed a *R* based tool *chipDyno* for the identification of quantitative prediction of regulatory activities of the gene specific TFA through posterior estimation. The *chipDynoUserGuide* attached at the end of this report explains different functionality of this tool and working pathway.

For *C. elegans* gene specific TFA is quite a new experiments. So we didn't find any other result to consider a baseline and compare our result.

According to WormNet [2014] the number of gene of *C. elegans* is 15,139 and Inmaculada et al. [2007] presented 934 transcription factors. All the network motif, i.e. autoregulation, multi-component loop, feedforward loop, single input, multi-input motif, regulator chain were visible for transcription factor activity. So it was a mammoth task to choose all the transcription factors and show their activity. Rather we choose some random transcription factor and tried to find out its activity on different genes.

As a random sample we choose transcription factor ZK370.2 and tried to find its activity on different genes. Figure 3.2 shows that ZK370.2 can act on C37C3.2, Y105E8B.3, Y45F10B.3, C34F11.3, F26E4.6 and T24G10.2. In the dataset we had three replication of same experimental setup and it outcome. We also tried to run our experiments three times and collect the result individually. Later we plot all the outcome together. From the outcome of our experimental result we can say that for some cases (i.e. F26E4.6 and T24G10.2) the result is very flat and doesn't looks so informative but some of the results represents TFAs varies notably over time (i.e. C37C3.2 and Y45F10B.3). Perhaps these are the genes which regulates significantly by this transcription factor. For some cases the error bar is quite high. False positive could be an issue here. The magnitude of TFA also differs from one to another. We picked another random transcription factor T20B12.8.3. Figure 3.3 shows its activity on different genes.

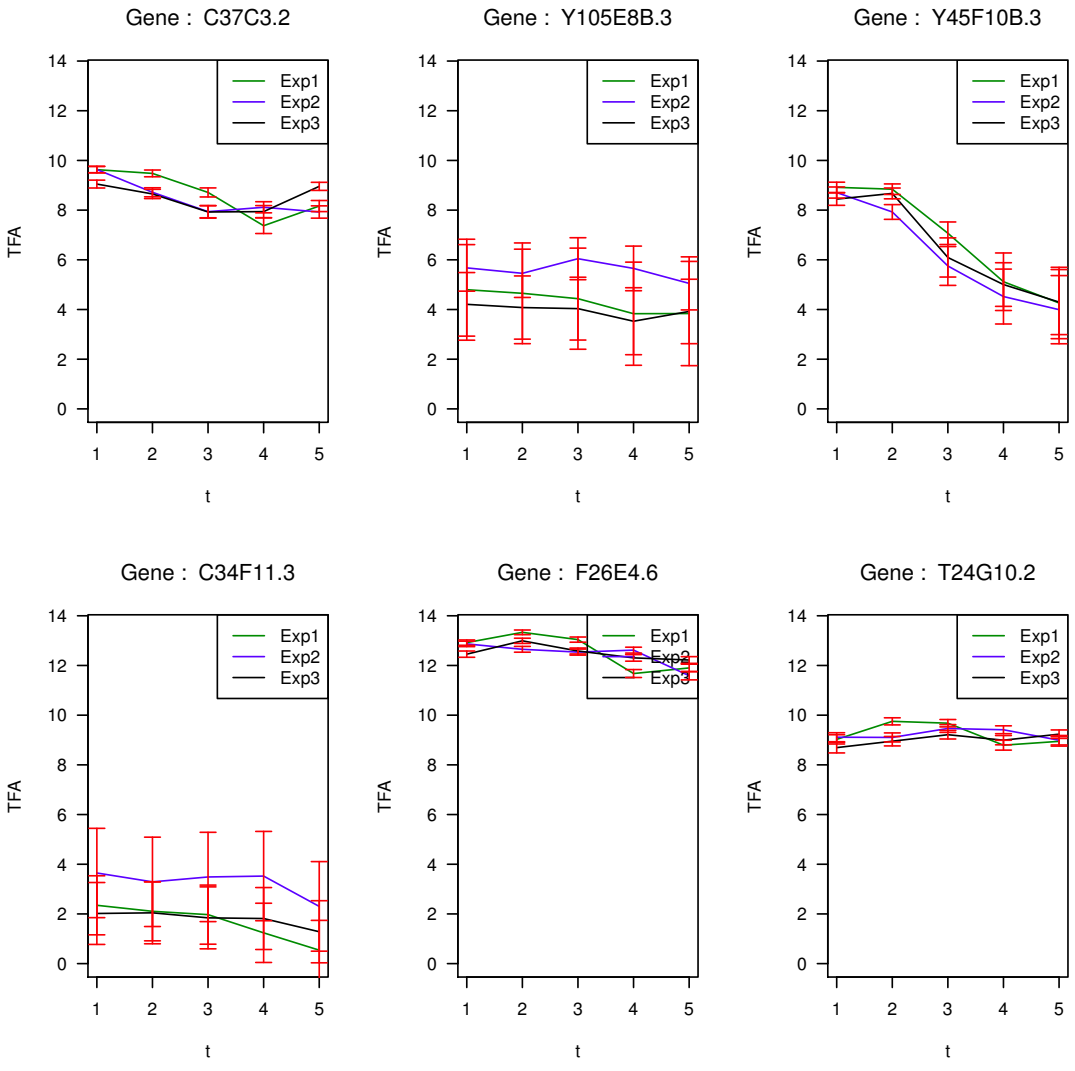


FIGURE 3.2: Gene Specific transcription factor activity of ZK370.2

3.3.1 Gene with multiple regulators

For the case of multi input motif a single gene could be regulated by multiple transcription factor. Our developed tool can determine the posteriori of the relative weight for the different transcription factors regulating the genes. Table 3.1 shows some examples. Gene C44B12.5 can be regulated by transcription factor Y116A8C.35 and F33A8.3. While gene Y105E8B.3 is regulated by T20B12.8, F33A8.3, Y116A8C.35, F11A10.2 and C16A3.7. Though for some cases the expression level is quite low and noise margin is significantly high but we can rank these gene using Kalaitzis and Lawrence [2011].

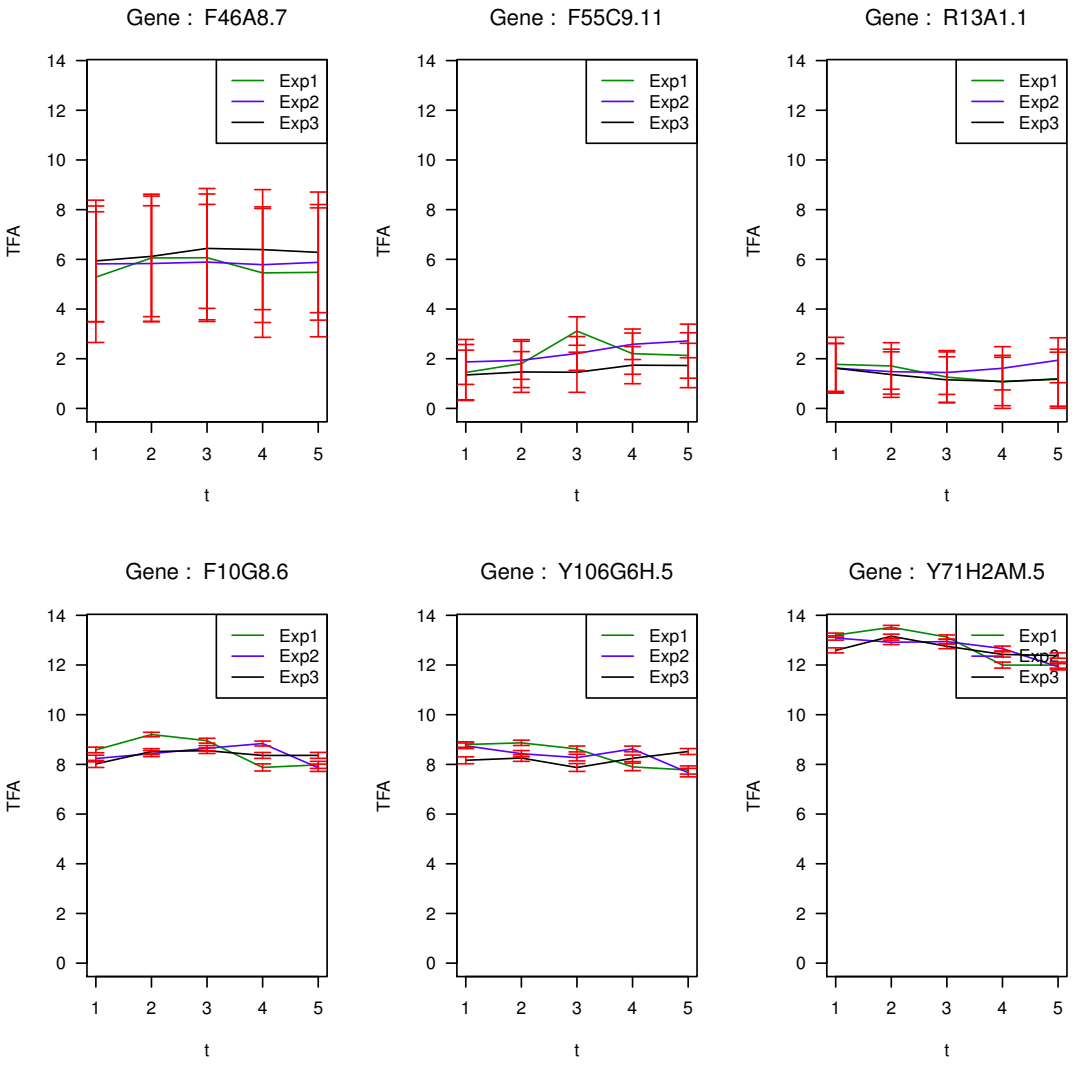


FIGURE 3.3: Gene Specific transcription factor activity of T20B12.8.3

Gene Name	Regulators activity
C44B12.5	Y116A8C.35 = 1.719797 ± 3.493205, F33A8.3 = 1.415785 ± 3.492985
Y105E8B.3	Y54G2A.1 = 0.07157665 ± 1.2222137 F33D11.12 = 0.03861905 ± 0.7252534 ZK370.2 = −1.20157055 ± 2.0318513
Y105E8B.3	T20B12.8 = 0.25474933 ± 2.5665869 F33A8.3 = 0.11619828 ± 3.5107742 Y116A8C.35 = 0.03289664 ± 3.8071374 F11A10.2 = 0.03016348 ± 1.7737585 C16A3.7 = 0.01883489± 0.9431105

TABLE 3.1: Genes regulated by multiple TF

3.3.2 Different clusters and corresponding active TF

Cossins et al. [2007] was also trying to some clusters analysis of the genes based on different phenotype and its subsequent activities of the cell properties[1]. Here are the basic clusters and some of the phenotype properties:

Cluster 1 - Chill upregulated basically related with cell morphogenesis, cell growth, regulation of cell size, electron transport regulation of cell growth, generation of precursor metabolites and energy, anatomical structure morphogenesis, cellular metabolic process, proteolysis, etc.

Cluster 2 - Chill late upregulated related with chromosome organization and biogenesis, DNA packaging, chromatin architecture chromatin modification, negative regulation of developmental process, chromatin remodelling, regulation of developmental process, DNA metabolic process larval development (sensu Nematoda), organelle organization and biogenesis, post-embryonic development etc.

Cluster 3 - Chill downregulated genes related with amino acid and derivative metabolic process, carboxylic acid metabolic process, organic acid metabolic process, fatty acid metabolic process, amino acid metabolic process, monocarboxylic acid metabolic process, etc. Rest of the genes were placed in the group 'Others'.

Figure 3.4 shows heat map generated from DNA microarray data to reflect the gene expression values at different temperature and their basic clusters Cossins et al. [2007][2]. Based on the above clusters we have tried to find out the the transcription factors active

[1]The result wasn't published but we came to know about it through some discussions.
[2]Some portion of the result is unpublished result; came to know through a presentation

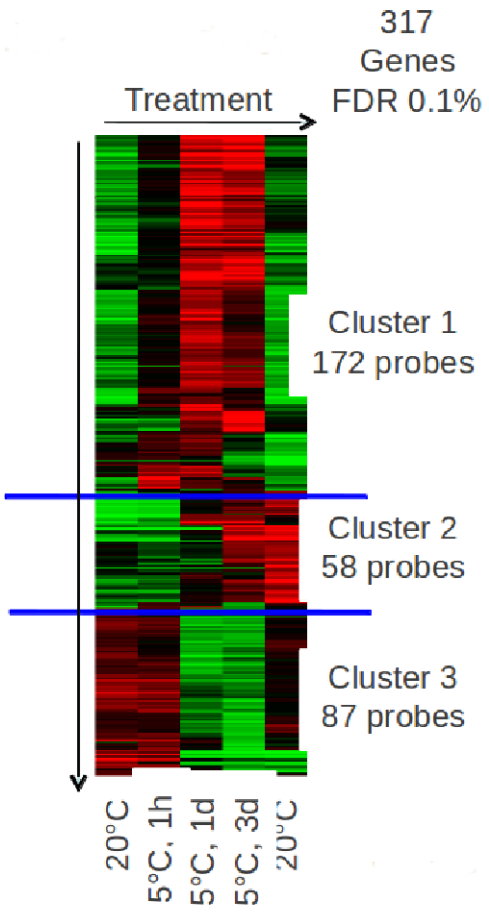


FIGURE 3.4: Clustering of TF

Clusters	Active TF
1. Chill upregulated	6
2. Chill late upregulated	245
3. Chill downregulated	128
4. Others	203

TABLE 3.2: Active TF on different clusters

for different clusters. We have managed to find out the active transcription factor for each clusters and Table 3.2 shows the numbers.

3.4 Ranking Differentially expressed gene expressions

Kalaitzis and Lawrence [2011] analysed the time series gene expression and filter the quiet or inactive genes from the differentially expressed genes. They have developed the model considering the temporal nature of data using Gaussian process. We have

used this model to rank our time series gene expression and ranked the differentially expressed gene expressions. We tried to rank the three replicates of our data separately and later determine the Pearson correlation between ranking score of different samples.

Figure 3.5 shows the Pearson correlation between different ranking scores. The correlation coefficient for all three relations (between sample 1 and sample 2, sample 2 and sample 3 and sample 3 and sample 1) were quite high. Which indicates the similarity of differentially expressed genes and quiet genes of different samples or replication of time series data. So, if required, based on these ranking we can easily filter out some of the quiet genes and keep the other genes for further experiments.

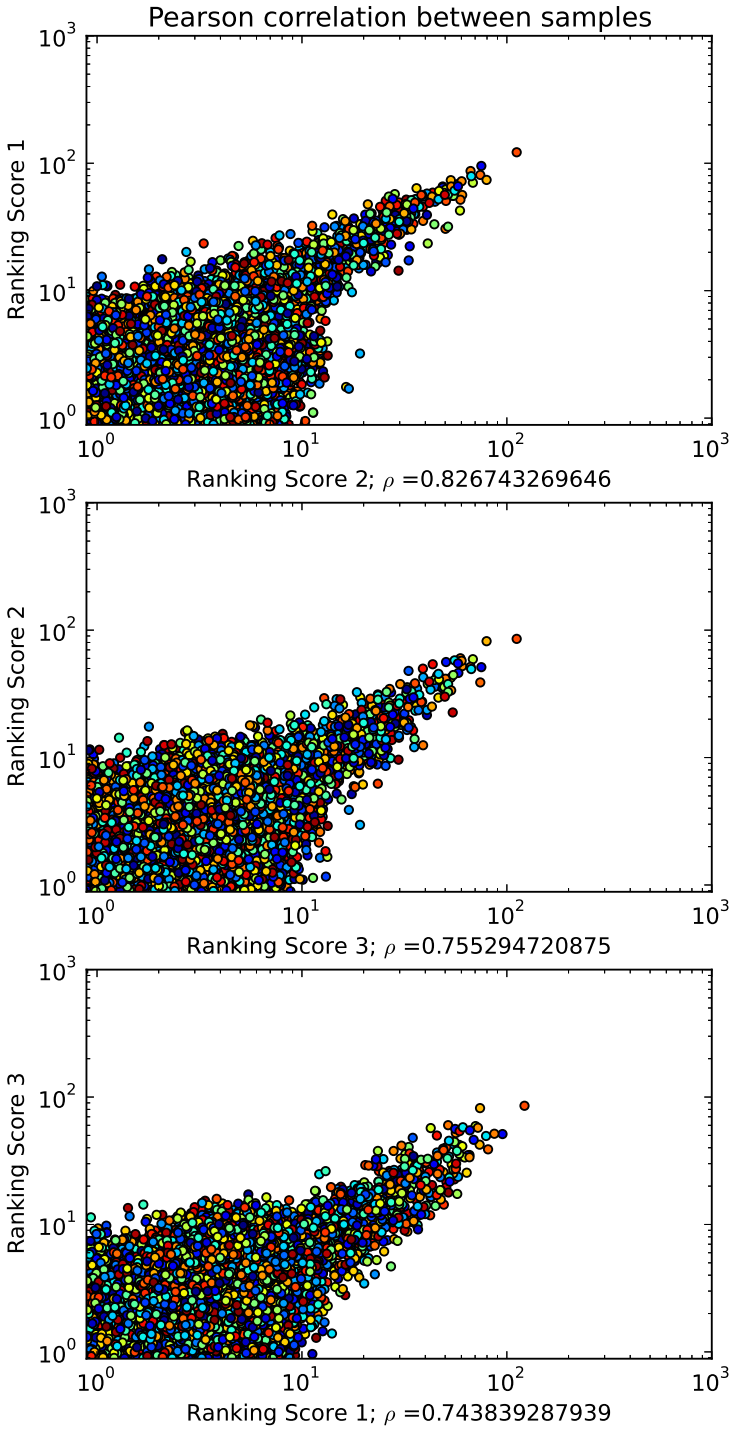


FIGURE 3.5: Pearson's correlation between different ranking scores

Chapter 4

Gaussian Process Regression

4.1 Brief History of Gaussian Process

The Gaussian processes is one of the most simple and widely used families of stochastic processes for modeling dependent data observed over time, or space, or time and space together. As a general setting, Gaussian process of many types have been studied and incorporated in research for decades. The Wiener process (e.g. Papoulis [1991]) (one of the best known Lévy processes) is a particular type of Gaussian process. The story of using Gaussian process is still a long one. Kolmogorov [1941] and Wiener [1949] used Gaussian process for time series prediction date backs to the 1940's. But probably the history of Gaussian process is even older. The Brownian motion is a Gaussian process. This is because the distribution of a random vector is a linear combination of vector which have a normal distribution (Castaneda et al. [2012]). Thorvald N. Thiele was the first to propose the mathematical theory of Brownian motion. He also introduce the likelihood function during the period 1860-1870 when he was serving as a assistant to professor H. L. d'Arrest at the Copenhagen Observatory, Denmark.

Since the 1970's Gaussian process have been widely adopted in the field of meteorology and geostatistics. Around that time Gaussian process regression was named as kriging and used by Matheron [1973] for prediction in geostatistics. O'Hagan [1978] used Gaussian process in the field of statistics for multivariate input regression problem. For general purpose function approximators Bishop [1995] used neural networks, Neal [1996] showed the link between Gaussian process and neural networks and in the machine learning context Williams and Rasmussen [1996] first described Gaussian process regression.

Over the last two decades Gaussian process in machine learning has turned to a major interest and much work has been done. Rasmussen and Williams [2006] perhaps the

most widely used and cited article on Gaussian process for machine learning and most of the discussed in this chapter can be found there in detailed form.

4.2 The Regression Problem

Machine learning problems can be roughly categorized into three basic classes.

1. Supervised learning: inferring a function from labelled training data
2. Unsupervised learning: to find hidden structure of unlabelled data
3. reinforcement learning: take action by maximizing the cumulative reward.

MacKay [2003], Bishop [2006] describes the concepts in detail. Supervised learning may be further sub-categorized in two fundamental tasks: regression and classification. Regression problem deals with estimating the relationship among some dependent variables with some independent variables, whereas classification identifies the desired discrete output levels.

Regression is the task of making some prediction of a continuous output variable at a desired input, based on a training input output data set. The input data can be any type of object or real valued features located in $\mathbb{R}^D$ which have some predictability for an unobserved location.

By definition of regression, it is obvious that there will be some inference based on a function mapping the outputs from a set of given inputs, because by inferring a function we can predict the response for a desired input. In the case of Bayesian inference, a prior distribution over function is required. Then the model go through some training process and update the prior, based on the training data set $\mathcal{D}$ constructed with N input vectors, such as $\{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X} \equiv \{\mathbf{x}_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^D$ are the training inputs and $\mathbf{y} \equiv \{y_n\}_{n=1}^N$, $\mathbf{y}_i \in \mathbb{R}$ are the training outputs. Now a key question arises, how can we consider a distribution over an infinite dimensional object as a function?

Although using plain and simple statistics regression problem can be solved, but to model a more complex and specific learning task with improved reliability and robustness Gaussian process is a better selection. Gaussian process models can be used for regression model having an object featuring infinite dimensionality. Even at present Gaussian process have been advanced beyond the regression model and now using for classification, unsupervised learningcite, reinforcement learning cite and many more.

We assume the outputs considered at the training level may contain some noise and observed from the underlying mapping function $f(\mathbf{x})$. The objective of the regression problem is to construct $f(\mathbf{x})$ from the data $\mathcal{D}$. This task is ill-defined and dealing with noisy data leads to an exercise in reasoning under uncertainty. Hence, a single estimate of $f(\mathbf{x})$ clearly could be misleading, rather a probability distribution over likely functions could be much more appealing. A regression model based on Gaussian process is a fully probabilistic Bayesian model, and definitely will serve for our purpose. In contrast with other regression models, here we will get the opportunity to choose the best estimate of $f(\mathbf{x})$. If we consider a probability distribution on functions $p(f)$ as the Bayesian prior for regression, then from data Bayesian inference can be used to make predictions:

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})} \tag{4.1}$$

The dynamic activity of transcription factors can be viewed as a regression task.

4.3 Gaussian Process definition

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen and Williams [2006]). It is a continuous stochastic process and defines probability distributions for functions. It can be also viewed as a random variables indexed by a continuous variable: $f(\mathbf{x})$ chosen from a random function variables $\mathbf{f} = \{f_1, f_2, f_3, ..., f_N\}$, with corresponding indexed inputs $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N\}$. In Gaussian processes, variables from these random functions are normally distributed and as a whole can be represent as a multivariate Gaussian distribution:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{4.2}$$

where $\boldsymbol{\mu}$ is the mean and $\mathbf{K}$ is covariance of Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ The Gaussian distribution is over vectors but the Gaussian process is over functions.

We need to define the mean function and covariance function for a Gaussian process prior. If $f(\mathbf{x})$ is a real process, a Gaussian process is completely defined by its mean function and covariance function given in 4.3 and 4.4 respectively. Usually the $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ are defined as-

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{4.3}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \tag{4.4}$$

where $\mathbb{E}$ represents the expected value. We denote the Gaussian process as-

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{4.5}$$

The covariance matrix $\mathbf{K}$ is constructed from the covariance function $k(\mathbf{x}, \mathbf{x}')$ and $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

4.4 GP: Covariances

For convenience, we often define the mean of the prior of the GP as zero but the posterior mean of the GP $p(f|\mathcal{D})$ obtained from the GP regression is not a zero mean process.

Based on our problem we are free to design our covariance function. The mandatory requirement of a covariance matrix is symmetric positive semi-definite. So as long as the covariance function generates symmetric positive semi-definite matrix, we can use that function for a Gaussian process. Smoothness, periodicity, amplitude, lengthscale etc. are the basic properties while choosing a Gaussian process covariance function. It is very crucial to choose a appropriate function for further Gaussian process Modelling. The main goal of this thesis is to develop a covariance function able to solve our problem, hopefully more robust and flexible way. Here first we will discuss about some of the very well known and widely used covariance functions. The in detail description will be found at Rasmussen and Williams [2006].

4.4.1 Exponentiated Quadratic covariance function

Exponentiated Quadratic covariance is the most widely used covariance function for Gaussian process. This is also known as squared exponential (SE) covariance or radial basis function (RBF). The exponentiated quadratic has become the de-facto default kernel for Gaussian process and has the following form-

$$K_{EQ}(r) = a^2 \exp\left(-\frac{r^2}{2l^2}\right) \tag{4.6}$$

where $r = \|\mathbf{x} - \mathbf{x}'\|$. Here $\|\mathbf{x} - \mathbf{x}'\|$ is invariant to translation and rotation. So, Exponentiated Quadratic covariance is stationary, as well as isotropic. Here the parameter for output variance a and lengthscale parameter l govern the property of sample functions and commonly known as hyperparameters. Parameter a determines the typical amplitude, i.e. average distance of the function away from the mean. l controls the lengthscale, i.e. the length of the wiggles of the function.

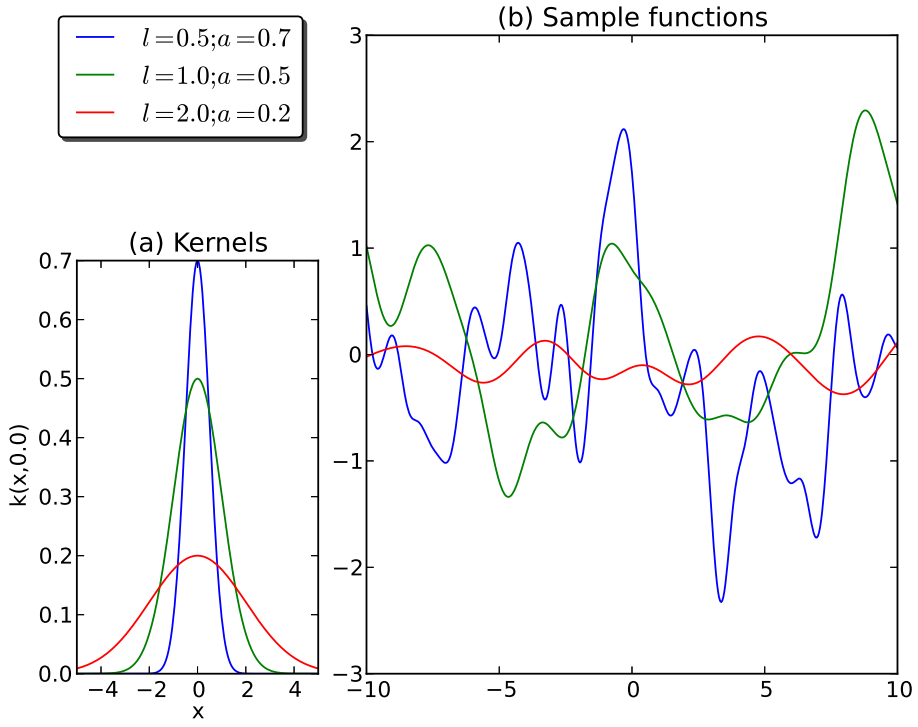


FIGURE 4.1: Exponentiated Quadratic kernel and sample functions

Figure 4.1(a) represents the kernel and Figure 4.1(b) shows random sample functions drawn from Gaussian process using Exponentiated Quadratic covariance with different lengthscale and amplitude hyperparameter. The random function was generated for a given input range by drawing a sample from the multivariate Gaussian using equation 4.2 with zero mean. The smoothness of the sample function depends on the equation 4.6. Function variable located closer in the input space are highly correlated, whereas function variable located at distance are loosely correlated or even uncorrelated. Exponentiated Quadratic covariance might be too smooth to perform any realistic regression task. Depending on the basic nature of the function other covariance function could be interesting.

4.4.2 Rational Quadratic covariance function

Rational Quadratic covariance function is equivalent to adding together multiple exponentiated quadratic covariance function having different lengthscale. Gaussian process prior kernel function expect smooth function with many lengthscale. Here the parameter α can control the relative weights for lengthscale variations. Exponentiated quadratic covariance function can be viewed as a special case of rational quadratic covariance

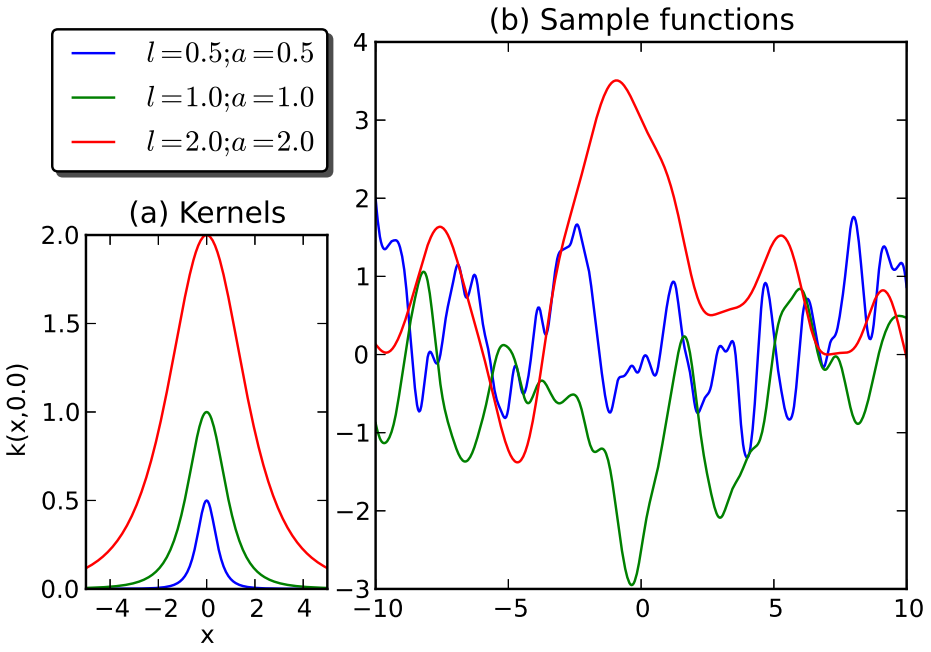


FIGURE 4.2: Rational Quadratic kernel and random sample functions

function. If $\alpha \to \infty$, then both of the functions become identical.

$$K_{RQ}(r) = a^2 \left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha} \tag{4.7}$$

where $r = \|\mathbf{x} - \mathbf{x}'\|$. Figure 4.2 (a) shows the kernels and (b) shows three different random sample functions drawn with different setting of hyperparameters a and l.

4.4.3 The Matérn covariance function

The Matérn class of covariance function are given by equation 4.8-

$$K_{Mat}(r) = a^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{l}\right) \tag{4.8}$$

where a, l, ν are positive hyperparameter, K_ν is a modified Bessel function and $\Gamma(.)$ is the Gamma function. Hyperparameter ν controls the roughness of the function and as like Exponentiated quadratic covariance function the parameters a and l controls the amplitude and lengthscale respectively. Though for $\nu \to \infty$ we can obtain the exponentiated quadratic kernel, but for finite value of ν the sample functions are significantly rough. The simpler form of Matérn covariance function is obtained when ν is half integer: $\nu = p + 1/2$, where p is a non-negative integer. The covariance function can be

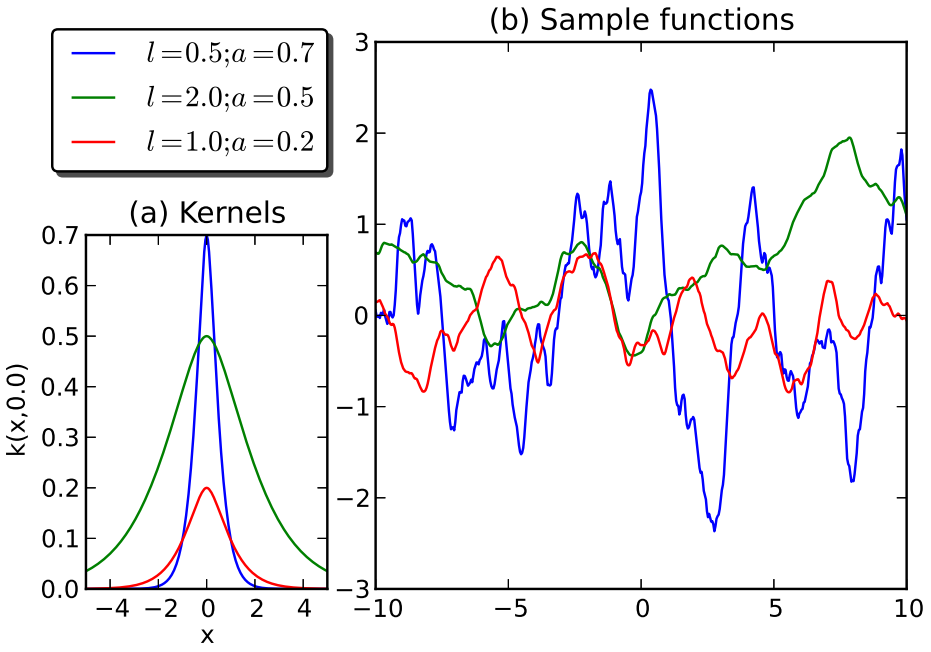


FIGURE 4.3: The Matérn32 kernel and random sample functions

expressed as a product of an exponential and a polynomial of order p. Abramowitz and Stegun [1965] derived the general expression as follows-

$$K_{\nu=p+1/2}(r) = \exp\left(-\frac{\sqrt{2\nu}r}{l}\right)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!\,(p-i)!}\left(\frac{\sqrt{8\nu}r}{l}\right)^{p-i} \tag{4.9}$$

The most interesting cases for machine learning are $\nu = 3/2$ and $\nu = 5/2$, for which we get the following equations respectively-

$$K_{\nu=3/2}(r) = \left(1+\frac{\sqrt{3}r}{l}\right)\exp\left(-\frac{\sqrt{3}r}{l}\right) \tag{4.10}$$

$$K_{\nu=5/2}(r) = \left(1+\frac{\sqrt{5}r}{l}+\frac{5r^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}r}{l}\right) \tag{4.11}$$

4.4.4 The Ornstein-Uhlenbeck Process

The Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein [1930]) is a special case of Matérn class covariance functions. The Ornstein-Uhlenbeck process was developed as a mathematical model of the velocity of a particle moving with Brownian motion. The OU process can be found setting up $\nu = 1/2$ and expressed as Equation 4.12. Figure 4.4(a) shows the kernel and Figure 4.4(b) shows the sample functions form the OU process

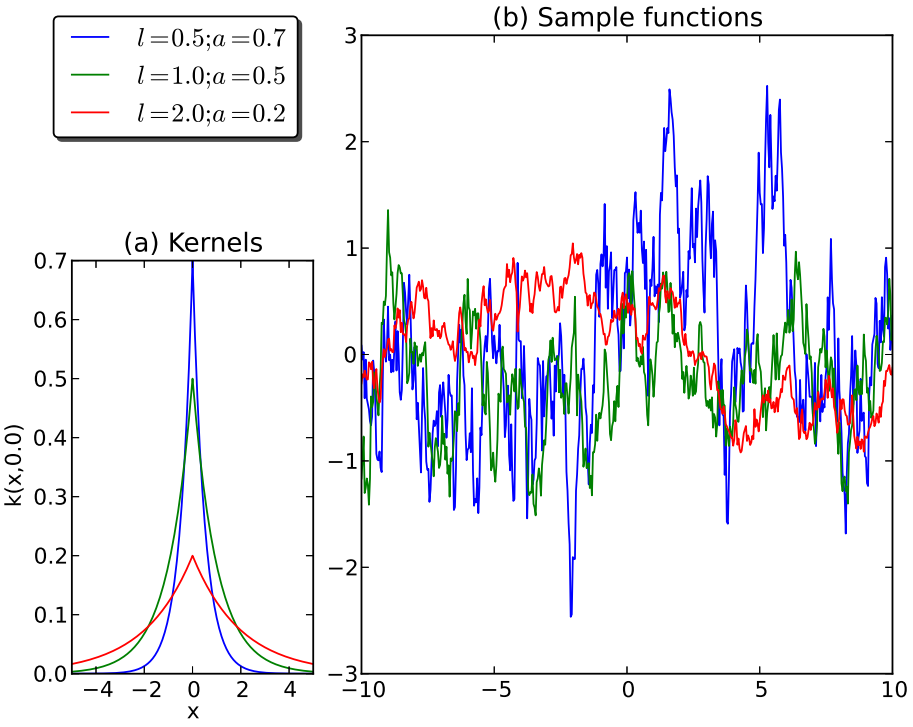


FIGURE 4.4: The OU kernel and random sample functions

having the exactly same amplitude parameter a and lengthscale parameter l.

$$K_{\nu=1/2}(r) = \exp\left(-\frac{r}{l}\right) \tag{4.12}$$

Figure 4.5 shows examples of some basic kernels- (a). Exponentiated Quadratic kernel, (b). Matérn52 kernel (c). Ornstein-Uhlenbeck kernel (d). Cosine kernel. These kernels are the realization of Exponentiated Quadratic covariance function(Equation 4.6), Matérn52 covariance function (Equation 4.11), Ornstein-Uhlenbeck covariance function (Equation 4.12) and Cosine function[1] respectively.

4.5 Gaussian Process Regression

Gaussian process regression can be done using the marginal and conditional properties of multivariate Gaussian distribution. Lets consider that we have some observations $\mathbf{f}$ of a function at observation point $\mathbf{x}$. Now we wish to predict the values of that function

[1]We have not described the Cosine covariance function here. The in detail description will be found at Rasmussen and Williams [2006]

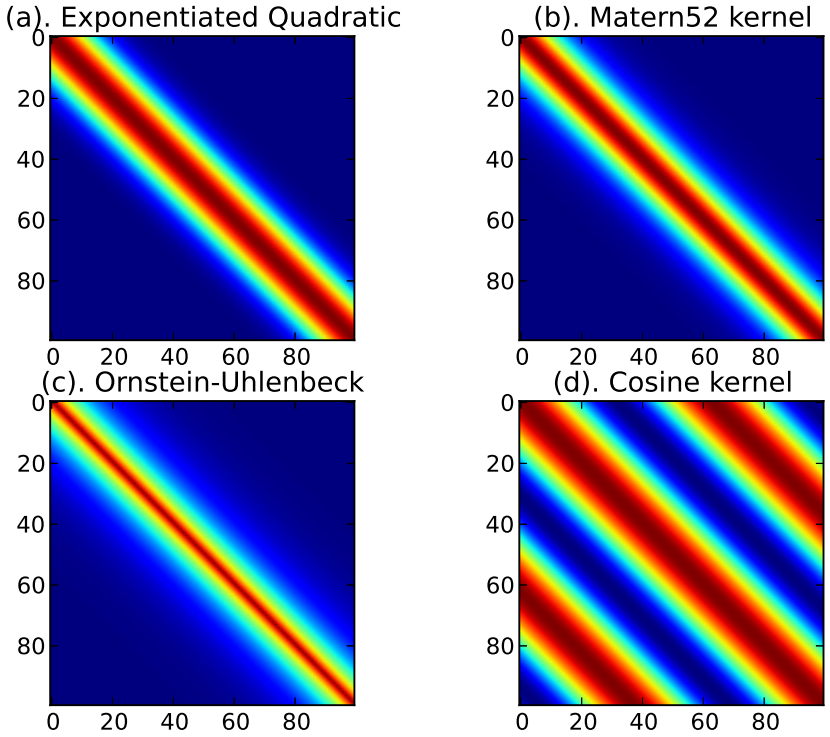


FIGURE 4.5: Representation of some basic kernels (a). Exponentiated Quadratic kernel, (b). Matérn52 kernel (c). Ornstein-Uhlenbeck kernel (d). Cosine kernel

at observation points $\mathbf{x}_\star$, which we are representing by $\mathbf{f}_\star$. Then the joint probability of $\mathbf{f}$ and $\mathbf{f}_\star$ can be obtained from equation 4.13-

$$p\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_\star\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_\star\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{K}_{\mathbf{x},\mathbf{x}} & \mathbf{K}_{\mathbf{x},\mathbf{x}_\star}\\ \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}} & \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}_\star}\end{bmatrix}\right) \tag{4.13}$$

where the covariance matrix $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ has elements derived from the covariance function $k(x, x')$, such that the $(i, j)^{th}$ element of $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ is given by $k(\mathbf{x}[i], \mathbf{x}[i])$ The conditional property of a multivariate Gaussian is used to perform regression the. The conditional property is can be represented by the equation 4.14:

$$p(\mathbf{f}|\mathbf{f}_\star) = \mathcal{N}\left(\mathbf{f}_\star\middle|\mathbf{K}_{\mathbf{x}_\star,\mathbf{x}}\mathbf{K}_{\mathbf{x},\mathbf{x}}^{-1}\mathbf{f}, \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}_\star} - \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}}\mathbf{K}_{\mathbf{x},\mathbf{x}}^{-1}\mathbf{K}_{\mathbf{x},\mathbf{x}_\star}\right) \tag{4.14}$$

In ideal case the observations $\mathbf{f}$ is noise free but in practice it is always corrupted with some noise. Lets consider $\mathbf{y}$ is the corrupted version of $\mathbf{f}$. If we consider this noise as Gaussian noise then we can write $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}\middle|\mathbf{f}, \sigma^2\mathbf{I}\right)$, where σ^2 is the variance of the

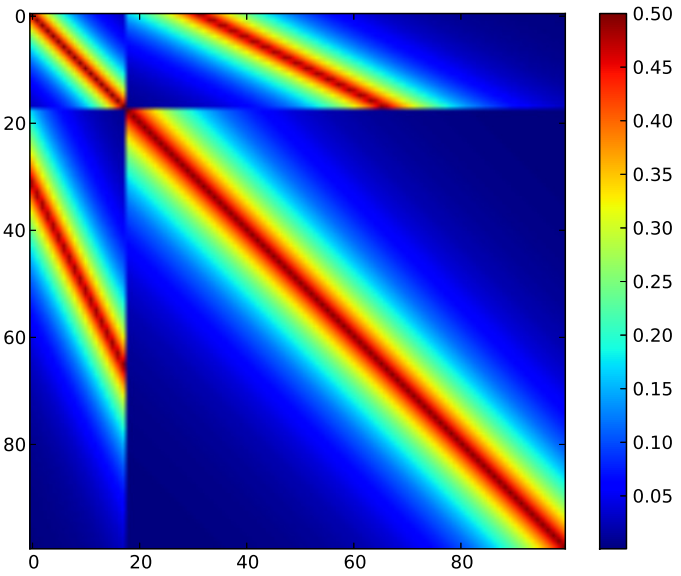


FIGURE 4.6: Overall representation of covariances between training and test data

noise and $\mathbf{I}$ is the identity matrix with appropriate size and marginalise the observation $\mathbf{f}$. Then the joint probability of $\mathbf{y}$ and $\mathbf{f}_\star$ can be represented by the equation 4.15.

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_\star\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_\star\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2\mathbf{I} & \mathbf{K}_{\mathbf{x},\mathbf{x}_\star}\\ \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}} & \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}_\star}\end{bmatrix}\right) \tag{4.15}$$

Regression with Gaussian process is Bayesian method. From the knowledge of a *prior* over a function we proceed to a *posterior* and this happens in a closed from of equation 4.14.

Figure 4.6 shows the overall covariance structure between some training and test data. For this example we choose 18 training points and 82 test points. We observed the shaded structure because some of the training data are closer to some of the test data. Observing this structure we can also figure out the closeness between training and test data.

4.5.1 Making prediction

The probability density is represented by functions. Due to consistency this density is known as a process. Also by this property, any future values of $\mathbf{f}_\star$ which are unobserved can be predicted without affecting $\mathbf{f}$. To make prediction of the test data we

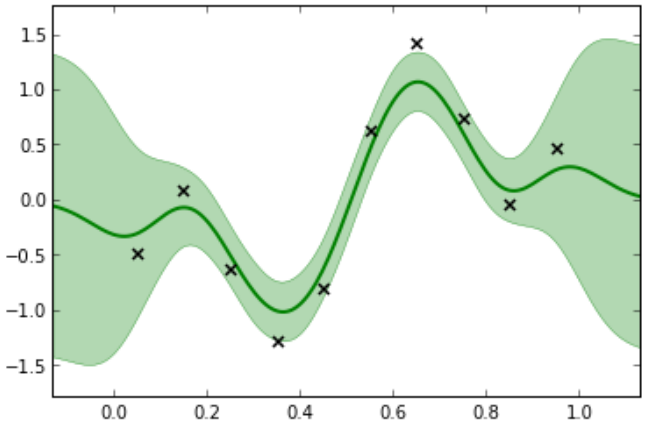


FIGURE 4.7: Simple example of regression using Gaussian process

use the conditional distribution. In ideal case the conditional distribution is $p(\mathbf{f}_\star|\mathbf{f})$ and if we consider the noise then the conditional distribution will be $p(\mathbf{f}_\star|\mathbf{y})$. Both of the distribution are also Gaussian,

$$\mathbf{f}_\star \sim (\boldsymbol{\mu}_f, \mathbf{C}_f) \tag{4.16}$$

The mean of the conditional distribution of Equation 4.16 is:

$$\boldsymbol{\mu}_f = \mathbf{K}_{\mathbf{x},\mathbf{x}_\star}^{\mathbf{T}} \left[\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{y} \tag{4.17}$$

and the covariance of the conditional distribution of Equation 4.16 given by:

$$\mathbf{C}_f = \mathbf{K}_{\mathbf{x}_\star,\mathbf{x}_\star} - \mathbf{K}_{\mathbf{x},\mathbf{x}_\star}^{\mathbf{T}} \left[\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{K}_{\mathbf{x},\mathbf{x}_\star} \tag{4.18}$$

These results can be calculated using block matrix inverse rules. The derivation can found in appendix section (Appendix **??**). Figure 4.7 shows a simple example of regression using Gaussian process.

4.5.2 Hyperparameter Learning

To construct the covariance function still we need to consider the hyperparameters and optimize those. The most efficient and commonly used optimization technique for hyperparameters can be done using maximum likelihood. If we consider all the hyperparameters α, σ^2 and l in to a vector $\boldsymbol{\theta}$, then we can use gradient methods to optimize $p(\mathbf{y}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. The Log likelihood is given by:

$$p\left(\mathbf{y}|\boldsymbol{\theta}\right) = -\frac{D}{2}log2\pi - \frac{1}{2} \times log\left|\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2\mathbf{I}\right| - \frac{1}{2}\mathbf{y}^T\left[\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2\mathbf{I}\right]^{-1}\mathbf{y} \tag{4.19}$$

We can have the Log maximum likelihood by:

$$\boldsymbol{\theta}_{max} = argmax\left(p\left(\mathbf{y}|\boldsymbol{\theta}\right)\right) \tag{4.20}$$

4.6 Toward the GP model of TFA

Simo Särkkä indicated an analogical pathway [2] to construct a kernel function for Gaussian process from Markovian assumption based probabilistic approach of Sanguinetti et al. [2006]. In the earlier probabilistic approach gene specific TFAs was obtained from-

$$\mathbf{b}_{n(t+1)} \sim \mathcal{N}(\gamma\mathbf{b}_{nt} + (1-\gamma)\boldsymbol{\mu}, (1-\gamma^2)\boldsymbol{\Sigma}) \tag{4.21}$$

For a discrete time variable k the above equation can be rewrite as-

$$\mathbf{b}_{n(k+1)} \sim \mathcal{N}\left(\gamma\mathbf{b}_{nk} + (1-\gamma)\boldsymbol{\mu}, (1-\gamma^2)\boldsymbol{\Sigma}\right), \tag{4.22}$$

and

$$\mathbf{b}_{n_1} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \tag{4.23}$$

Let's now form a continuous model which has these same finite-dimensional distributions. First construct a one-dimensional process with the property-

$$u_{k+1} \sim \mathcal{N}\left(\gamma u_k + (1-\gamma)\,\mu, (1-\gamma^2)s\right), \tag{4.24}$$

where μ and s are scalar.

We can now assume that u_k's are actually values u_{t_k} from a continuous process $u(t)$ and let's assume that-

$$t_k = kDt. \tag{4.25}$$

A good candidate for this kind of model is the mean-reverting $Ornstein - Uhlenbeck$ model (Uhlenbeck and Ornstein [1930])-

$$du = -\lambda\left(u-\mu\right)dt + q^{1/2}dB, \tag{4.26}$$

[2] Not published yet, came through some discussions

where B is a standard Brownian motion (i.e., Wiener process). This equation can now be solved on the time instants t_k and the result is a recursion

$$u(t_k) = au(t_{k-1}) + b\mu + w_{k-1}, \tag{4.27}$$

where $w_{k-1} \sim \mathcal{N}(0, c)$ with-
$a = \exp(-\lambda Dt)$

$b = \int_0^D t \exp(-\lambda(Dt - s))ds$
$= 1 - \exp(-\lambda Dt)$

$c = \int_0^D t \exp(-\lambda(Dt - s))q \exp(-\lambda(Dt - s))ds$
$= q \int_0^D t \exp(-2\lambda(Dt - s))ds$
$= [q/(2\lambda)][1 - \exp(-2\lambda Dt)]$

That is,

$$u_{k+1} \sim \mathcal{N}\left(au_k + b\mu, c\right). \tag{4.28}$$

We can now match the coefficients:

$$a = \exp(-\lambda Dt) = \gamma \tag{4.29}$$

$$b = 1 - \exp(-\lambda Dt) = 1 - \gamma \tag{4.30}$$

$$c = (1 - \gamma^2)s = [q/(2\lambda)][1 - \exp(-2\lambda Dt)] \tag{4.31}$$

Equation 4.29 quite luckily has a nice solution $\gamma = \exp(-\lambda Dt)$ and from Equation 4.31 we will have another solution $s = q/(2\lambda)$, which can be inverted to give $\lambda = -[1/Dt] \log \gamma$ and $q = -[2s/Dt] \log \gamma$.

If we arbitrarily fix $Dt = 1$, we get $\lambda = -\log \gamma$
$q = -2s \log \gamma$.

We can now recall that the (stationary) covariance function of the Ornstein-Uhlenbeck process we get-

$k_u(t, t') = [q/(2\lambda)] \exp(-\lambda|t - t'|)$
$= s \exp((\log \gamma)|t - t'|)$
$= s \exp(|t - t'|(\log \gamma))$
$= s \exp(\log \gamma^{|t-t'|})$
$= s\gamma^{|t-t'|}$.

When we start from variance $s = q/\left[2\lambda\right]$, then the process will indeed be stationary from the start. Returning to the original vector valued $\mathbf{b}$, because the system is separable, we can conclude that the implied covariance function is just obtained by formally replacing s with $\mathbf{\Sigma}$ everywhere-

$$\mathbf{K}_b(t, t') = \mathbf{\Sigma}\boldsymbol{\gamma}^{|t-t'|} \tag{4.32}$$

Thus is equivalent to considering the vector process of mean-reverting $Ornstein - Uhlenbeck$ model

$$\mathbf{db} = -\lambda(\mathbf{b} - \boldsymbol{\mu})\mathbf{dt} + Q^{1/2}\mathbf{dB}. \tag{4.33}$$

Chapter 5

Gaussian Process Model of Gene Expressions

In this chapter we design a covariance function for reconstructing transcription factor activities given gene expression profiles and a connectivity matrix (binding data) between genes and transcription factors. Our modelling framework builds on ideas of Sanguinetti et al. [2006] who used a linear-Gaussian state-space modelling framework to infer the transcription factor activity of a group of genes.

We note that the linear Gaussian model is equivalent to a Gaussian process with a particular covariance function. We therefore build a model directly from the Gaussian process perspective to achieve the same effect. We introduce a computational trick, based on judicious application of singular value decomposition, to enable us to efficiently fit the Gaussian process in a reduced 'TF activity' space.

First we load in the classic Spellman et al. [1998] Yeast Cell Cycle data set. The cdc15 time series data has 23 time points. We can load this gene expression data in with GPy.

Time series of synchronized yeast cells from the CDC-15 experiment of Spellman et al. [1998]. Two colour spotted cDNA array data set of a series of experiments to identify which genes in Yeast are cell cycle regulated. We can make a simple helper function to plot genes from the data set (which are provided as a pandas array).

Our second data set is from ChiP-chip experiments performed on yeast by Lee et al. [2002]. These give us the binding information between transcription factors and genes. In this notebook we are going to try and combine this binding information with the gene expression information to infer transcription factor activities.

5.1 Model for Transcription Factor Activities

We are working with *log* expression levels in a matrix $\mathbf{Y} \in \Re^{n \times T}$ and we will assume a linear (additive) model giving the relationship between the expression level of the gene and the corresponding transcription factor activity which are unobserved, but we represent by a matrix $\mathbf{F} \in \Re^{q \times T}$. Our basic assumption is as follows. Transcription factors are in time series, so they are likely to be temporally smooth. Further we assume that the transcription factors are potentially correlated with one another (to account for transcription factors that operate in unison).

Correlation Between Transcription Factors: If there are q transcription factors then the correlation between different transcription factors is encoded in a covariance matrix, $\mathbf{\Sigma}$ which is $q \times q$ in dimensionality.

Temporal Smoothness: Further we assume that the log of the transcription factors' activities is temporally smooth, and drawn from an underlying Gaussian process with covariance $\mathbf{K}_t$.

Intrinsic Coregionalization Model: We assume that the joint process across all q transcription factor activities and across all time points is well represented by an intrinsic model of coregionalization where the covariance is given by the Kronecker product of these terms.

$$\mathbf{K}_f = \mathbf{K}_t \otimes \mathbf{\Sigma} \tag{5.1}$$

This is known as an intrinsic coregionalization model (Wackernagel [2003]). Alvarez et al. [2012] presented the machine learning orientated review of these methods. The matrix $\mathbf{\Sigma}$ is known as the coregionalization matrix.

5.2 Relation to Gene Expressions

We now assume that the jth gene's expression is given by the product of the transcription factors that bind to that gene. Because we are working in log space, that implies a log linear relationship. At the ith time point, the log of the jth gene's expression, $\mathbf{y}_{i,j}$ is linearly related to the log of the transcription factor activities at the corresponding time point, $\mathbf{f}_{i,:}$. This relationship is given by the binding information from $\mathbf{S}$. We then assume that there is some corrupting Gaussian noise to give us the final observation.

$$\mathbf{y}_{i,j} = \mathbf{S}\mathbf{f}_{:,i} + \boldsymbol{\epsilon}_i \tag{5.2}$$

where the Gaussian noise is sampled from

$$\epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{5.3}$$

5.3 Gaussian Process Model of Gene Expression

We consider a vector operator which takes all the separate time series in $\mathbf{Y}$ and stacks the time series to form a new vector $n \times T$ length vector $\mathbf{y}$. A similar operation is applied to form a $q \times T$ length vector $\mathbf{f}$. Using Kronecker products we can now represent the relationship between $\mathbf{y}$ and $\mathbf{f}$ as follows: Standard properties of multivariate Gaussian distributions tell us that

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}), \tag{5.4}$$

where

$$\mathbf{K} = \mathbf{K}_t \otimes \mathbf{S}\mathbf{\Sigma}\mathbf{S}^\top + \sigma^2\mathbf{I}. \tag{5.5}$$

This results in a covariance function that is of size n by T where n is number of genes and T is number of time points. However, we can get a drastic reduction in the size of the covariance function by considering the singular value decomposition of $\mathbf{S}$. The matrix $\mathbf{S}$ is n by q matrix, where q is the number of transcription factors. It contains a 1 if a given transcription factor binds to a given gene, and zero otherwise. The likelihood of a multivariate Gaussian is:

$$L = -\frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\mathbf{y}^\top\mathbf{K}^{-1}\mathbf{y} \tag{5.6}$$

In the worst case, because the vector $\mathbf{y}$ contains $T \times n$ points (T time points for each of n genes) we are faced with $O(T^3n^3)$ computational complexity. We are going to use a rotation trick to get the likelihood.

5.4 The Main Computational Trick

5.4.1 Rotating the Basis of a Multivariate Gaussian

For any multivariate Gaussian you can rotate the data set and compute a new rotated covariance which is valid for the rotated data set. Mathematically this works by first inserting $\mathbf{R}\mathbf{R}^\top$ into the likelihood at three points as follows:

$$L = -\frac{1}{2}\log|\mathbf{K}\mathbf{R}^\top\mathbf{R}| - \frac{1}{2}\mathbf{y}^\top\mathbf{R}^\top\mathbf{R}\mathbf{K}^{-1}\mathbf{R}^\top\mathbf{R}\mathbf{y} + \text{const} \tag{5.7}$$

The rules of determinants and a transformation of the data allows us to rewrite the likelihood as

$$L = -\frac{1}{2}\log|\mathbf{R}^\top\mathbf{K}\mathbf{R}| - \frac{1}{2}\hat{\mathbf{y}}^\top\left[\mathbf{R}^\top\mathbf{K}\mathbf{R}\right]^{-1}\hat{\mathbf{y}} + \text{const} \tag{5.8}$$

where we have introduced the rotated data: $\hat{\mathbf{y}} = \mathbf{R}\mathbf{y}$. Geometrically what this says is that if we want to maintain the same likelihood, then when we rotate our data set by $\mathbf{R}$ we need to rotate either side of the covariance matrix by $\mathbf{R}$, which makes perfect sense when we recall the properties of the multivariate Gaussian.

5.4.2 A Kronecker Rotation

In this paper we are using a particular structure of covariance which involves a Kronecker product. The rotation we consider will be a Kronecker rotation (Stegle et al. [2011]). We are going to try and take advantage of the fact that the matrix $\mathbf{S}$ is square meaning that $\mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^\top$ is not full rank (it has rank of most q, but is size $n \times n$, and we expect number of transcription factors q to be less than number of genes n).

When ranks are involved, it is always a good idea to look at singular value decompositions (SVDs). The SVD of $\mathbf{S}$ is given by:

$$\mathbf{S} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{V}^\top \tag{5.9}$$

where $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$, $\boldsymbol{\Lambda}$ is a diagonal matrix of positive values, $\mathbf{Q}$ is a matrix of size $n \times q$: it matches the dimensionality of $\mathbf{S}$, but we have $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$. Note that because it is not square, $\mathbf{Q}$ is not in itself a rotation matrix. However it could be seen as the first q columns of an n dimensional rotation matrix (assuming n is larger than q, i.e. there are more genes than transcription factors).

If we call the $n - q$ missing columns of this rotation matrix $\mathbf{U}$ then we have a valid rotation matrix $\mathbf{R} = \begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}$. Although this rotation matrix is only rotating across the n dimensions of the genes, not the additional dimensions across time. In other words we are choosing $\mathbf{K}_t$ to be unrotated. To represent this properly for our covariance we need to set $\mathbf{R} = \mathbf{I} \otimes \begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}$. This gives us a structure that when applied to a covariance of the form $\mathbf{K}_t \otimes \mathbf{K}_n$ it will rotate $\mathbf{K}_n$ whilst leaving $\mathbf{K}_t$ untouched.

When we apply this rotation matrix to $\mathbf{K}$ we have to consider two terms, the rotation of $\mathbf{K}_t \otimes \mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^\top$, and the rotation of $\sigma^2\mathbf{I}$.

Rotating the latter is easy, because it is just the identity multiplied by a scalar so it remains unchanged

$$\mathbf{R}^\top\mathbf{I}\sigma^2\mathbf{R} = \mathbf{I}\sigma^2 \tag{5.10}$$

The former is slightly more involved, for that term we have

$$\left[\mathbf{I}\otimes\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}^\top\right]\mathbf{K}_t\otimes\mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^\top\left[\mathbf{I}\otimes\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}\right]=\mathbf{K}_t\otimes\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}^\top\mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^\top\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}. \quad (5.11)$$

Since $\mathbf{S}=\mathbf{Q}\boldsymbol{\Lambda}\mathbf{V}^\top$ then we have

$$\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}^\top\mathbf{S}\boldsymbol{\Sigma}\mathbf{S}^\top\begin{bmatrix}\mathbf{Q} & \mathbf{U}\end{bmatrix}=\begin{bmatrix}\boldsymbol{\Lambda}\mathbf{V}^\top\boldsymbol{\Sigma}\mathbf{V}\boldsymbol{\Lambda} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}. \quad (5.12)$$

This prompts us to split our vector $\hat{\mathbf{y}}$ into a q dimensional vector $\hat{\mathbf{y}}_u=\mathbf{U}^\top\mathbf{y}$ and an $n-q$ dimensional vector $\hat{\mathbf{y}}_q=\mathbf{Q}^\top\mathbf{y}$. The Gaussian likelihood can be written as

$$L=L_u+L_q+\text{const} \quad (5.13)$$

where

$$L_q=-\frac{1}{2}\log|\mathbf{K}_t\otimes\boldsymbol{\Lambda}\mathbf{V}^\top\boldsymbol{\Sigma}\mathbf{V}\boldsymbol{\Lambda}+\sigma^2\mathbf{I}|-\frac{1}{2}\hat{\mathbf{y}}_q^\top\left[\mathbf{K}_t\otimes\boldsymbol{\Lambda}\mathbf{V}^\top\boldsymbol{\Sigma}\mathbf{V}\boldsymbol{\Lambda}+\sigma^2\mathbf{I}\right]^{-1}\hat{\mathbf{y}}_q \quad (5.14)$$

and

$$L_u=-\frac{T(n-q)}{2}\log\sigma^2-\frac{1}{2\sigma^2}\hat{\mathbf{y}}_u^\top\hat{\mathbf{y}}_u \quad (5.15)$$

Strictly speaking we should fit these models jointly, but for the purposes of illustration we will firstly use a simple procedure. Firstly, we fit the noise variance σ^2 on $\hat{\mathbf{y}}_u$ alone using L_u. Once this is done, fix the value of σ^2 in L_q and optimize with respect to the other parameters.

With the current design the model is switching off the temporal correlation. The next step in the analysis will be to reimplement the same model as described by Sanguinetti et al. [2006] and recover their results. That will involve using an Ornstein Uhlenbeck covariance and joint maximisation of the likelihood of L_u and L_q.

Exponentiated Quadratic kernel is very smooth kernel compared to Ornstein-Uhlenbeck kernel and perhaps is not a very good choice for the determination of actual transcription factors activities. Still it can figure out the basic nature of the activities with over smoothness. Figure 5.1 shows activities of different transcription factors while the model was developed considering Exponentiated Quadratic kernel with White kernel in additive form.

Figure 5.2[1] shows transcription factor activity of ACE2. While developing the model we choose Ornstein-Uhlenbeck kernel and White kernel in additive form. We believe that

[1]The complete model is still in progress. After the final model exact figure might be updated from this one

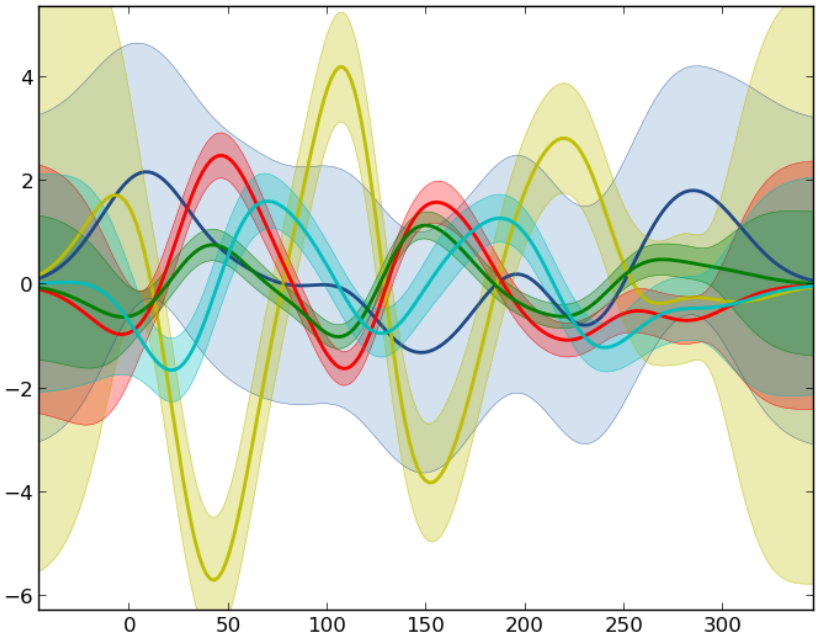


FIGURE 5.1: Variation of activities of Transcription factors RBF+white kernels

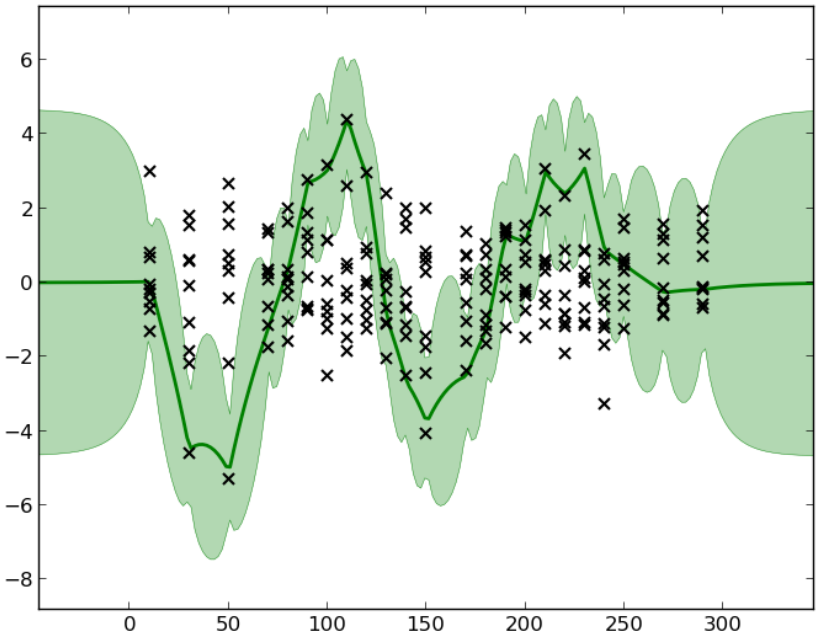


FIGURE 5.2: Transcription factor activity of ACE2

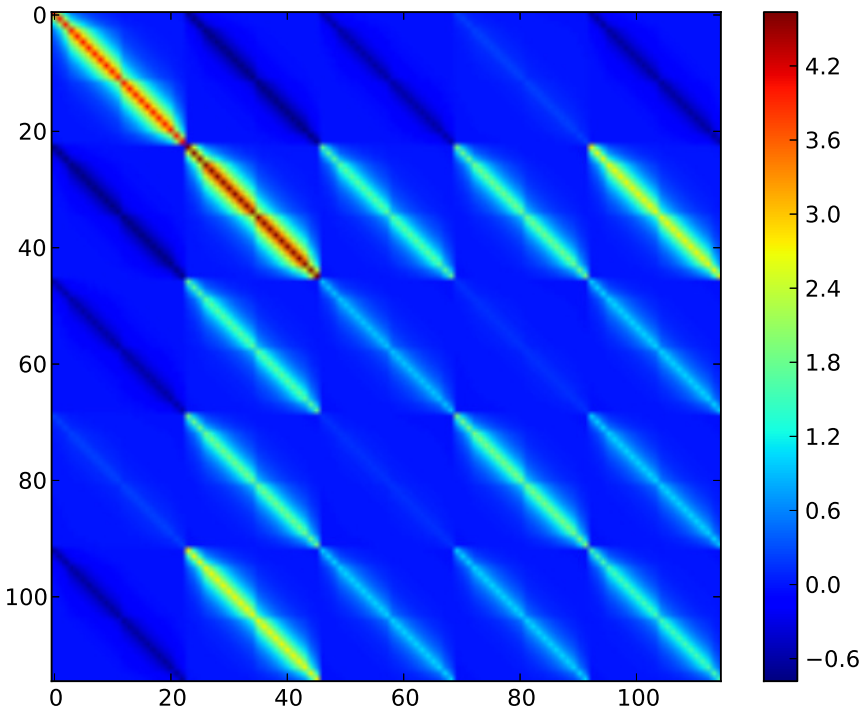


FIGURE 5.3: Kernel of Intrinsic Coregionalization model $\mathbf{K}_f$ considering 6 Transcription factors where covariance matrix $\mathbf{\Sigma}$ of (*Equation*5.5) was constructed using Ornstein-Uhlenbeck kernel and White kernel in additive form

the Ornstein-Uhlenbeck kernel will consider the basic nature of the transcription factors activity while White kernel will deal the noise associated the collected gene expression data.

Figure 5.3 shows the pictorial representation of intrinsic coregionalization kernel (Equation 5.5) $\mathbf{K}_f$ considering 20 transcription factors where covariance matrix $\mathbf{\Sigma}$ of was constructed using Ornstein-Uhlenbeck kernel and White kernel in additive form.

Figure 5.4 shows some examples of transcription factors activities where model was developed with Ornstein-Uhlenbeck kernel and White kernel.

5.5 Making prediction

Using Kronecker product we can rewrite the Equation 5.4 as:

$$\mathbf{y_q} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} + \sigma^2 \mathbf{I}\right) \tag{5.16}$$

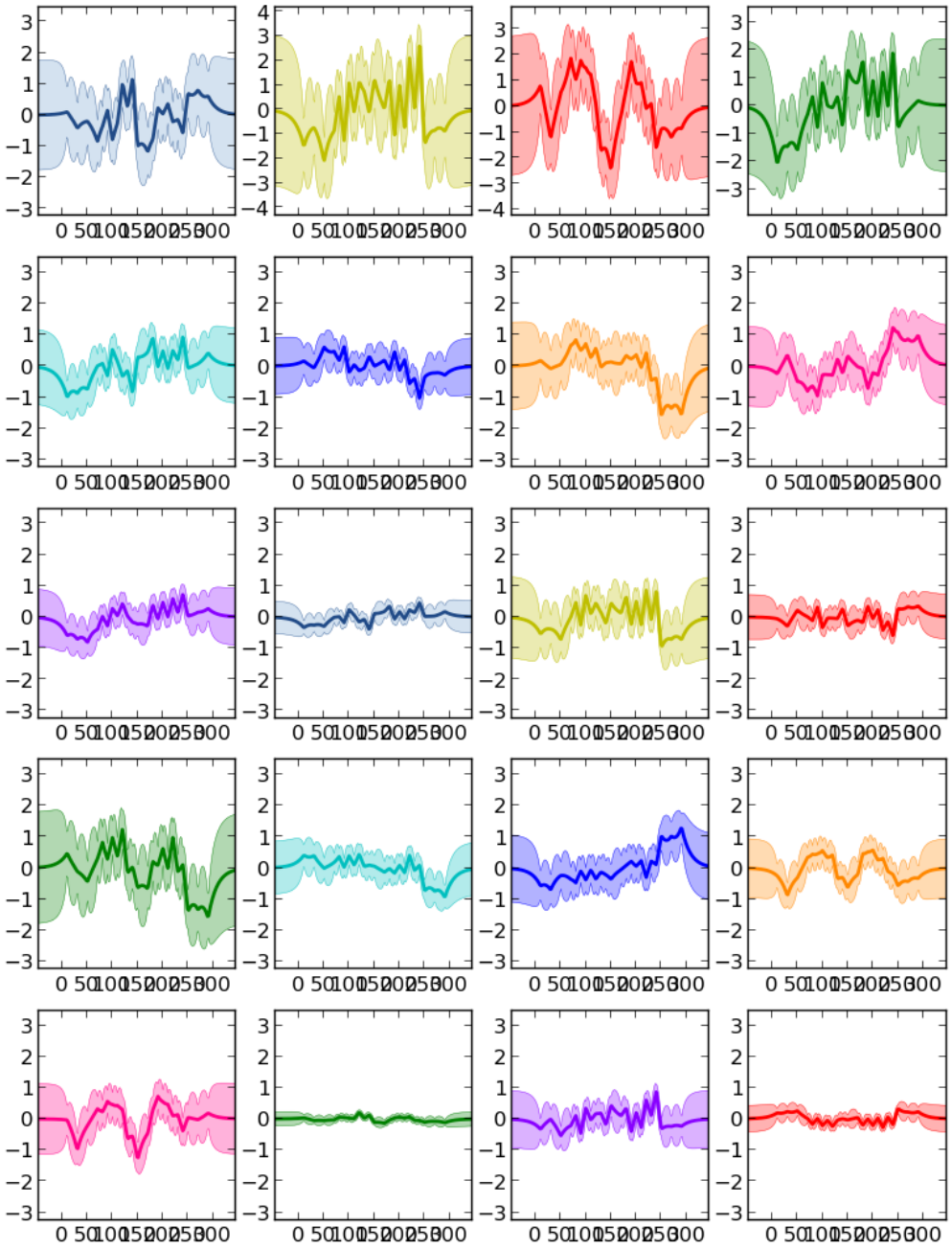


FIGURE 5.4: Transcription factor activity of different TF using Ornstein-Uhlenbeck kernel and White kernel in additive form

Standard properties of multivariate Gaussian distributions tells us can split equation 5.16 into

$$\mathbf{y_q} = \mathbf{g} + \boldsymbol{\epsilon} \tag{5.17}$$

where $\mathbf{g}$ and $\boldsymbol{\epsilon}$ are also Gaussian distributions and can be represented by:

$$\mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda}\right) \tag{5.18}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right) \tag{5.19}$$

Now we can represent the matrix $\mathbf{F}$ of transcription factor activity as:

$$\mathbf{F} = \mathbf{I} \otimes \mathbf{V}\Lambda^{-1}\mathbf{g} \tag{5.20}$$

$$\mathbf{\Sigma} = \mathbf{W}\mathbf{W}^T + diag\left(\boldsymbol{\kappa}\right) \tag{5.21}$$

where $\boldsymbol{\kappa}$ is the kappa value from coregionalization matrix.

$$\mathbf{F} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_{t,t} \otimes \mathbf{\Sigma}\right) \tag{5.22}$$

Now we can find the conditional distribution of g for given y_q by:

$$p\left(\mathbf{g}|\mathbf{y}_q\right) \sim \mathcal{N}\left(\boldsymbol{\mu}_g, \mathbf{C}_g\right) \tag{5.23}$$

with a mean given by:

$$\boldsymbol{\mu}_g = \left[\mathbf{K}_{t_\star,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda}\right] \left[\mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} + \sigma^2 \mathbf{I}\right]^{-1} \mathbf{y}_q \tag{5.24}$$

and the covariance given by:

$$\boldsymbol{C}_g = \left[\mathbf{K}_{t_\star,t_\star} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda}\right] - \left[\mathbf{K}_{t_\star,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} \left[\mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} + \sigma^2 \mathbf{I}\right]^{-1} \mathbf{K}_{t_\star,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda}\right] \tag{5.25}$$

The mean of the conditional distribution of Equation 5.16 is:

$$\boldsymbol{\mu}_F = \mathbf{K}_{t_\star,t} \otimes \mathbf{\Sigma V \Lambda} \left[\mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} + \sigma^2 \mathbf{I}\right]^{-1} \mathbf{y}_q \tag{5.26}$$

and the covariance of the conditional distribution of Equation 5.16 given by:

$$\boldsymbol{C}_F = \mathbf{K}_{t_\star,t_\star} \otimes \mathbf{\Sigma} - \mathbf{K}_{t_\star,t} \otimes \mathbf{\Sigma V \Lambda} \left[\mathbf{K}_{t,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma V \Lambda} + \sigma^2 \mathbf{I}\right]^{-1} \left[\mathbf{K}_{t_\star,t} \otimes \mathbf{\Lambda V}^T \mathbf{\Sigma}\right] \tag{5.27}$$

Chapter 6

Conclusion and Future work

We have developed a tool based on programming language *R* named *chipDyno* using the model of Sanguinetti et al. [2006] which integrate the connectivity information between genes and transcription factors, and micro array data. The probabilistic nature of the model can determine the significant regulations in a given experimental condition.

Earlier the model was developed for a unicellular microorganism (yeast) but we have successfully manage to determine the gene specific transcription factor activity for *C. elegans*, a multicellular eukaryote. We were also successful to filter out the quiet genes from the differentially expressed genes.

To elucidate pathways and processes relevant to human biology and disease *C. elegans* is been using as a vital model. Different orthology-prediction methods (Shaye and Greenwald [2011]) are using to compile a list of *C. elegans* orthologs of human genes. Already a list of 7,663 unique protein-coding genes were resulted in that list and this represents 38% of the 20,250 protein-coding genes predicted in *C. elegans*. When human genes introduced into *C.Elegans* human genes replaced their homologous. On the contrary, many *C. elegans* genes can function with great deal of similarity to human like mammalian genes. So, the biological insight acquire from *C. elegans* may be directly applicable to more complex organism like human.

Lots of computational approaches on gene expression data for time series analysis are not well suited where time points are irregularly spaced. Even in commonly used state-space model time points must occur at regular intervals. On the other side gene expression experiments with regular samples may not be cost effective or optimal from the perspective of statistics. It is expected that models with irregular time points might be more informative if the time points are selected considering some temporal features. Gaussian process is not restricted to equally spaced time series data. Already Gaussian

process regression have been successfully applied to overcome this issue and analyse time series data (Kalaitzis and Lawrence [2011]). So our expected model will overcome the restriction of temporal sampling of equally spaced time intervals.

6.1 Future Work

Sanguinetti et al. [2006] model to infer the transcription factor activity is a linear-Gaussian state-space model. We believe that this linear Gaussian model is equivalent to Gaussian process with a specific covariance function. We have developed a model directly from Gaussian process to achieve the same goal. We are quite close to develop a valid covariance function for reconstructing transcription factor activities given gene expression profile and binding information between genes and transcription factors. Here we will introduce a computational trick using singular value decomposition and intrinsic coregionalization model. We believe this method will enable us to efficiently fit the Gaussian process in a reduced transcription factor activity space.

Amyotrophic lateral sclerosis (ALS), also known as "Lou Gehrig's Disease" or motor neurone disease (MND), is an irreversible progressive neurodegenerative adult onset that affects motor neurons in the brain and the spinal cord. Muscle denervation spreads over neuromuscular system and leads toward death by failure of the respiratory system with in few years of system onset (Peviani et al. [2010]). This lethal invariable disorder has median survival of less than 5 years, only 20% of the affected people can survive more than 5 years and 10% of the patients can survive more than 10 years. Mutation in the Cu/Zn superoxide dismutase (SOD1) gene is responsible for around 20% of the familial motor neurone disease Nardo et al. [2013]. Transgenic mice can express human SOD1 mutation and nicely replicates different histopathological and clinical features of motor neurone disease. Mimic of these murine models of different clinical phenotypes observed from human MND patients are widely using by the researchers to determine the disease progression. But we didn't found any evidence of gene expression analysis that attempt to analyse motor neuron disease considering the genetic background on different phenotype of this murine disease. So gene expression analysis for different murine models could reveal interesting information. Nardo et al. [2013] used two mouse models to analyse fast and slow disease progression of ALS. We will use our Gaussian process based model to infer the transcription factor activity on gene expression data obtained from different murine models and try to find out some fascinating insights.

Clustering of gene expression time series is another major interest of the research to get the view of groups of co-regulated or associated genes. It is assumed that gene involved in the same biological process will be expressed with a similarity sharing underlying

time series. Cossins et al. [2007] did some additional cluster analysis (not published yet!) based on some phenotype properties. Again it is very common to have multiple biological replicates of the gene expression time series data. Just taking average of the replicates surely lead toward discarding insight. Recently Hensman et al. [2013] used a hierarchy of Gaussian process to model a gene specific and replicate specific temporal covariance. They also used this model for clustering application. Using this Gaussian process based hierarchical clustering analysis of Hensman et al. [2013] we will try to find some robust clusters for the gene expression data of *C. elegans*. Once if we can do so, it will easily lead us to find out the active transcription factors related with these clusters and their subsequent dynamic behaviour as well.

Appendix A

Summery of DDP Progress

Based on the plan of the Doctoral Development Programme (DDP) various activities have been completed-

A.1 Posters Presentation and talks

A.1.1 Posters

- Rahman M A, Lawrence N D, "A Probabilistic Dynamic Model for Transcription Factor Activity of C. Elegans", Machine Learning Summer School and AISTATS joint poster session, Reykjavic, Iceland 2014.

A.1.2 Workshop Notebook

- Rahman M A, Lawrence N D, "Inferring Transcription Factor Activities by Combining Binding Information with Gene Expression Profiles", Available online, url-http://nbviewer.ipython.org/github/SheffieldML/notebook/blob/master/compbio/transcriptionFactorActivities.ipynb

A.1.3 Talks

- Rahman M A, Lawrence N D, "Dynamic model for transcription factor activity of C. Elegans", Sheffield Institute for Translational Neuroscience (SITraN), University of Sheffield, November 2013.

- Rahman M A, Lawrence N D, "A probabilistic dynamic model for transcription factor activity of C. Elegans", Sheffield Institute for Translational Neuroscience (SITraN), University of Sheffield, May 2013.

A.2 Participation in Conference, Workshop and Summer schools

As a part of DDP progress I have participated a number of Conference, Workshop and Summer schools given bellow-

- Gaussian Process Summer School 2013, University of Sheffield, UK was organized by our lab ML@SITraN from June 10 to June 12, 2013 followed by Workshop on Latent Force Model 2013. I was an organizing member here.
- I have attended the Natural Computing Applications Forum (NCAF) Meeting at University of Oxford, Oxford, UK from July 04 to July 05, 2013.
- I have attend the Machine Learning Summer School (Advanced Topics in Machine Learning) at Technical University of Denmark, Copenhagen, Denmark on August 2013.
- Gaussian Process Winter School 2014, University of Sheffield, UK was organized by our lab ML@SITraN from January 13 to January 16, 2014. I was a organizing member here. Workshop on Spatiotemporal Modelling with Gaussian Processes was collocated with GPWS 2014.
- I have attended the Mathematical and Statistical Aspects of Molecular Biology (MASAMB) 2014 workshop at University of Sheffield, UK from April 09 to April 10, 2014
- I have attended the conference of Artificial Intelligence and Statistics (AISTATS) 2014 at Reykjavik, Iceland followed by the Machine Learning Summer School (MLSS2014).
- Gaussian Process Summer School 2014, University of Sheffield, UK was organized by our lab ML@SITraN from September 15 to September 17, 2014. Workshop on Gaussian Processes for Feature Extraction was collocated with GPSS 2014. I was a organizing member here.

A.3 Demonstration

As a part of DDP progress the activities relating to teaching and demonstrating are given bellow-

- COM4509/COM6509 Machine Learning and Adaptive Intelligence with Professor Neil D Lawrence(ACADEMIC YEAR 2014-15).
- COM161: Introduction to Programming and Problem-Solving and COM160: Computer Problem Solving and Object-Oriented Design with Professor Mark Hepple (ACADEMIC YEAR 2014-15).
- COM1005/COM2007: Machines and Intelligence with Professor Rob Gaizauskas (ACADEMIC YEAR 2014-15).
- FCE101: Introduction to Bioengineering (ACADEMIC YEAR 2011-12) with Professor Neil D lawrence.

A.4 Participation in DDP Courses

To improve the core concepts of research I have attended the following courses/modules:

- COM - COM4509 COM6509: Machine Learning and Adaptive Intelligence(GRADUATE YEAR 2011-12).
- COM4508/6508: Introduction to Computational Systems Biology (GRADUATE YEAR 2011-12).
- FCE - FCE6100: Research Ethics and Integrity (GRADUATE YEAR 2011-12).
- COM - COM6930: Research Teamwork in Computer Science (GRADUATE YEAR 2011-12).
- COM - COM6940: Presentation of Computer Science Research (GRADUATE YEAR 2011-12).
- COM - COM6950: Developments in Computer Science (GRADUATE YEAR 2011-12).

A.5 Other activities related to DDP

- Take part in the poster presentation at the research retreat of Department of Computer Science, University of Sheffield.
- Reading, understanding and writing about relevant research, and implementing simple algorithms to improved research related skills.
- Maintaining a personal website detailing any research and experiments that have been done.
- Made contact with experts from a different department/universities to get the in-depth knowledge on Machine Learning and System Biology.

Bibliography

Abramowitz, M. and Stegun, I. A. (1965). *Handbook of Mathematical Functions*. Dover, New York.

Alon, U. (2006). An introduction to systems biology: design principles of biological circuits. *CRC Press*.

Alter, O. and Golub, G. H. (2004). Integrative analysis of genome-scale data by using pseudoinverse projection predicts novel correlation between dna replication and rna transcription. *Proceedings of the National Academy of Sciences USA*, 101(47):16577–16582.

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3).

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

Boulesteix, A.-L. and Strimmer, K. (2005). Predicting transcription factor activities from combined analysis of microarray and chip data: a partial least squares approach. *Theoretical Biology and Medical Modelling*, 2(23).

Brenner, S. (1974). The genetics of caenorhabditis elegans. *Genetics*, 77:71–94.

Brivanlou, A. H. and Darnell, J. E. (2002). Transcription signal transduction and the control of gene expression. *Science*, 295(5556):819–818.

Byerly, L., Cassada, R., and Russell, R. (1976). The life cycle of the nematode caenorhabditis elegans. i. wild-type growth and reproduction. *Dev Biol*, 51(1):23–33.

Castaneda, L. B., Arunachalam, V., and Dharmaraja, S. (2012). *Introduction to Probability and Stochastic Processes with Applications*. Wiley.

Cossins, A. R., Murray, P., Hayward, S. A. L., Govan, G. G., and Gracey, A. Y. (2007). An explicit test of the phospholipid saturation hypothesis of acquired cold tolerance in caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 104(13):5489–5494.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *J. Comput. Biol.*, 7(3/4):601–620.

Gao, F., Foat, B. C., and Bussemaker, H. J. (2004). Defining transcriptional networks through integrative modeling of mrna expression and transcription factor binding data. *BMC Bioinformatics*, 5(31).

Gerstein, M. B., Lu, Z. J., Nostrand, E. L. V., and Cheng, C. (2010). Integrative analysis of the caenorhabditis elegans genome by the modencode project. *Science*, 330.

Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J.-B., Reynolds, D. B., Yoo1, J., Jennings, E. G., Zeitlinger, J., Pokholok, D. K., Kellis, M., Rolfe, P. A., Takusagawa, K. T., Lander, E. S., Gifford, D. K., Fraenkel, E., and Young, R. A. (2004). Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104.

Hensman, J., Lawrence, N. D., and Rattray, M. (2013). Hierarchical bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BMC Bioinformatics*, 14(252).

Ingalls, B. P. (2012). *Mathematical Modelling in Systems Biology: An Introduction*. MIT Press.

Inmaculada, B. M., Philippe, V., Fabien, C., Laurent, J., and Albertha, W. (2007). Edgedb: a transcription factor-dna interaction database for the analysis of c. elegans differential gene expression. *BMC Genomics*, 8(1):21.

Kalaitzis, A. A. and Lawrence, N. D. (2011). A simple approach to ranking differentially expressed gene expression time courses through gaussian process regression. *BMC Bioinformatics*, 12(180).

Karin, M. (1990). Too many transcription factors: positive and negative interactions. *New Biol.*, 2(2):126–131.

Kitano, H. (2000). Perspectives on systems biology. *New Gen. Comput.*, 18(3):199–216.

Kitano, H. (2002). *Systems Biology: Toward System-level Understanding of Biological Systems*. MIT Press.

Kolmogorov, A. N. (1941). Interpolation und extrapolation von stationären zufälligen folgen. *Bull. Acad. Sci. (Nauk) U.R.S.S. Ser. Math.*, 5:3–14.

Latchman, D. S. (1997). Transcription factors: an overview. *Int. J. Biochem. Cell Biol.*, 29(12):1305–1312.

Lee, I., Lehner, B., Crombie, C., Wang, W., Fraser, A. G., and Marcotte, E. M. (2007). A single network comprising the majority of genes accurately predicts the phenotypic effects of gene perturbation in c. elegans. *Nature Genetics*, 40:181–188.

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. T., Thompson, C. M., Simon, I., Zeitlinger, J., Jennings, E. G., Murray, H. L., Gordon, D. B., Ren, B., Wyrick, J. J., Tagne, J.-B., Volkert, T. L., Fraenkel, E., Gifford, D. K., and Young, R. A. (2002). Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804.

Lee, T. I. and Young, R. A. (2000). Transcription of eukaryotic protein-coding genes. *Annu. Rev. Genet.*, 34:77–137.

Liao, J. C., Boscolo, R., Yang, Y.-L., Tran, L. M., Sabatti, C., and Roychowdhury, V. P. (2003). Network component analysis: Reconstruction of regulatory signals in biological systems. *PNAS*, 100(26).

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in Applied Probability*, 5:439–468.

Mitchell, P. J. and Tjian, R. (1989). Transcriptional regulation in mammalian cells by sequence-specific dna binding proteins. *Science*, 245(4916):371–378.

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

Nardo, G., Iennaco, R., Fusi, N., Heath, P. R., Marino, M., Trolese, M. C., Ferraiuolo, L., Lawrence, N., Shaw, P. J., and Bendotti, C. (2013). Transcriptomic indices of fast and slow disease progression in two mouse models of amyotrophic lateral sclerosis. *Brain A journal of Neurology*, 136(11).

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics No. 118, New York: Springer-Verlag.

Nikolov, D. B. and Burley, S. K. (1997). Rna polymerase ii transcription initiation: A structural view. *Proc. Natl. Acad. Sci. U.S.A.*, 94(1):15–22.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40(1):1–42.

Ong, I. M., Glasner, J. D., and Page, D. (2002). Modelling regulatory pathways in e. coli from time series expression profiles. *Oxford Univ Press*, 18(1):S241–S248.

Palikaras, K. and Tavernarakis, N. (2013). Caenorhabditis elegans (nematode). (1):404–408.

Papoulis, A. (1991). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Companies, 3rd edition.

Pearson, R. D., Liu, X., Sanguinetti, G., Milo, M., Lawrence, N. D., and Rattray, M. (2009). puma: a bioconductor package for propagating uncertainty in microarray analysis. *BMC Bioinformatics*, 10(211).

Peviani, M., Caron, I., Pizzasegola, C., Gensano, F., Tortarolo, M., and Bendotti, C. (2010). Unraveling the complexity of amyotrophic lateral sclerosis: recent advances from the transgenic mutant sod1 mice. *CNS Neurol Disord Drug Targets*, 9(4):491–503.

Ptashne, M. and Gann, A. (1997). Transcriptional activation by recruitment. *Nature*, 386:569–577.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*.

Roeder, R. G. (1996). The role of general initiation factors in transcription by rna polymerase ii". *Trends Biochem. Sci.*, 21(9):327–335.

Sanguinetti, G., Rattray, M., and Lawrence1, N. D. (2006). A probabilistic dynamical model for quantitative inference of the regulatory mechanism of transcription. *Bioinformatics, Oxford University Press*, 22(14):1753–1759.

Shaye, D. D. and Greenwald, I. (2011). Ortholist: A compendium of c. elegans genes with human orthologs. *PLoS ONE*, 9(1).

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

Stegle, O., Lippert, C., Mooij, J. M., Lawrence, N. D., and Borgwardt, K. M. (2011). Efficient inference in matrix-variate gaussian models with \iid observation noise. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 630–638. Curran Associates, Inc.

Sulston, J. E. and Horvitz, H. (1977). Post-embryonic cell lineage of the nematode, caenorhabditis elegans. *Developmental Biology*, 56(1):110–156.

Sulston, J. E., Schierenberg, E., j. G. White, and Thomson, J. N. (1980). The embryonic cell lineage of the nematode caenorhabditis elegans. *Developmental Biology*, 100(1):64–119.

Tu, B. P., Kudlicki, A., Rowicka, M., and McKnight, S. L. (2005). Logic of the yeast metabolic cycle: Temporal compartmentalization of cellular processes. *Science*, 310(5751):1152–1158.

Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Phys. Rev.*, 36:823–841.

Wackernagel, H. (2003). *Multivariate Geostatistics An Introduction with Applications*. Springer Science and Business Media.

Watson, J. D. and Crick, F. H. (1953). Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, (171):737–738.

Wiener, N. (1949). *Extrapolation, Interpolation and Smoothing of Stationary Time Series.*

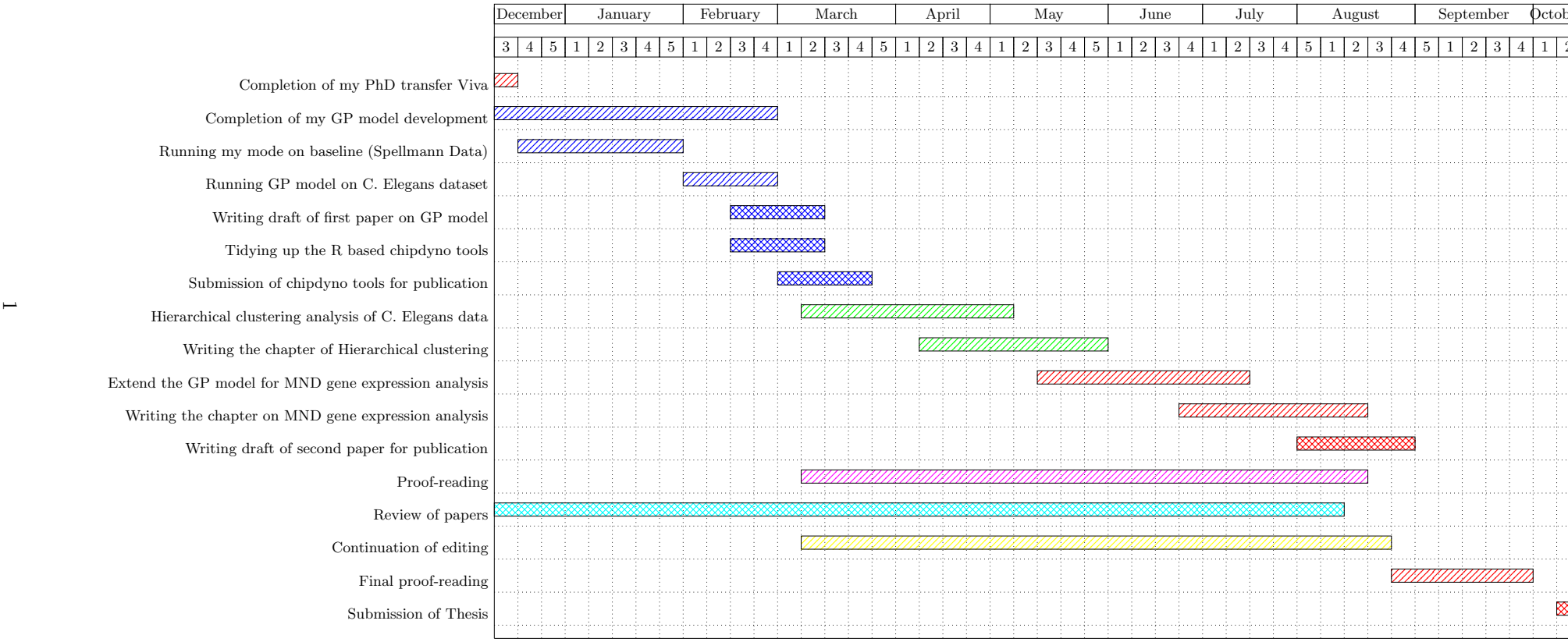
Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. *Advances in Neural Information Processing Systems 8, MIT Press*, pages 1–7.

Wood, W. B. (1988). The nematode c. elegans. *Cold Spring Harbor Laboratory Press, New York*, pages 1–16.

WormNet (2014). Wormnet. http://www.functionalnet.org/wormnet/about.html [Online; accessed 30-Nov-2014].

Xie, X., Lu, J., Kulbokas, E. J., Golub, T. R., Mootha, V., Lindblad-Toh, K., Lander, E. S., and Kellis, M. (2005). Systematic discovery of regulatory motifs in human promoters and 3' utrs by comparison of several mammals. *Nature*, (434):338–345.

Detailed work plan for completing my thesis



chipDyno User Guide

Muhammad A. Rahman, Guido Sanguinetti, Magnus Rattray and Neil D. Lawrence

November 3, 2014

Abstract

Quantitative estimation of the regulatory relationship between transcription factors and genes is a fundamental stepping stone when trying to develop models of cellular processes. This task, however, is difficult for a number of reasons: transcription factors' expression levels are often low and noisy, and many transcription factors are post-transcriptionally regulated. It is therefore useful to infer the activity of the transcription factors from the expression levels of their target genes.

Contents

1 Mirroring the chipDyno repository from GitHub

The recommended way to clone the package from GitHub. Installation should ensure that all the dependencies are met.

```
$ clone https://github.com/SheffieldML/chipDyno.git
$ cd chipDyno/
```

2 Installation from .tar file

After downloading the package it can also be installed in an specific location. If we want to use the directory $/data/Rpackages/$ then creating a package directory, the installation can be done in the following way:

```
$ sudo R
...
> install.packages("chipDynoTest2*", lib="/data/Rpackages/")
> library(chipDynoTest2*, lib.loc="/data/Rpackages/")
```

Or from the command line:

```
$ R CMD INSTALL chipDyno*.tar.gz
```

3 Loading the package and getting help

The first step in any *chipDyno* analysis is to load the package. This package can be loaded when the *R* console is ready. At the *R* console type the following command:

```
> library("chipDyno")
```

Command `help` can be used to get help on any function. For example, to get help on the `chipDynoLikeStatGrad` type the following (both of them have the similar output):

```
> help(chipDynoLikeStatGrad)
> ?chipDynoLikeStatGrad
```

4 Input data and preparations

If the data is in Affymetrix CEL files then it may required to do some preprocessing. This CEL files can be extracted using the bioconductor package *puma* [Pearson et al., 2009]. *puma* usually stores the point expression in '*_exprs.csv' file and defferent levels of uncertainties in '*_se.csv' files as Comma-separated values.

Load the point expression levels data and it corresponding uncertainty. Here the data was stored in a simple text file

```
> rm(list=ls())
> source("../chipDynoLoadModules.R")
> file_data <-
+   "../data/MetabolData/YeastMetabolism_exprs.txt";
> file_vars <-
+   "../data/MetabolData/YeastMetabolism_se.txt";
> data = as.matrix(read.table(file_data))
> data[1:5,1:8]
```

```
            V1        V2        V3        V4        V5         V6         V7         V8
[1,]  3.032196  3.915297  3.513140  2.887694  2.624707  2.4336185  0.5146789  3.0277957
[2,]  7.619968  7.009363  6.327303  6.420305  6.474659  6.4259952  5.9848784  6.6202407
[3,]  9.367415  9.337176  9.346070  9.528845  9.870827  9.9068906  9.2679571  9.7358944
[4,] 11.116018 10.157765 10.559681 10.625835 10.832905 11.1418061 10.7627800 11.1802351
[5,] -1.746557 -1.909657  1.074617 -2.313146 -1.508921 -0.9244119 -1.7330923 -0.8551462
```

```
> vars=as.matrix(read.table(file_vars))
> vars[1:5,1:8]
```

```
            V1        V2        V3        V4        V5         V6        V7         V8
[1,] 0.5988238 0.3692819 0.4940527 0.6459059 0.8544423 0.82382426 1.4128785 0.62460252
[2,] 0.1291985 0.1683352 0.2438547 0.2245224 0.2532125 0.23656530 0.2989101 0.21290733
[3,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.00000000
[4,] 0.1019400 0.1426169 0.1248059 0.1224968 0.1122887 0.09996799 0.1154624 0.09959699
[5,] 1.8253530 1.8705859 0.9995836 1.9992325 1.8095992 1.60575819 1.8439322 1.57351633
```

These differentially expressed data and its uncertainty doesn't contain any genes name or problem ID number. The gene Names and its corresponding problem IDs can be extracted from other two source (?) dictionary.txt and probeIDTu.txt.

```
> file_dictionary <-
+   "../data/MetabolData/dictionary.txt";
> dictionary <- read.table(file_dictionary, header = FALSE, sep = "\t",
+                     quote = "", dec = ".", col.names=c("IDs_temp", "ORF",
+                                  "ORF_temp","geneCN_temp"), na.strings = "NA",
+                                  colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
+                                  fill = TRUE, strip.white = FALSE, blank.lines.skip = FALSE,
+                                  comment.char = "#", allowEscapes = FALSE, flush = FALSE)
> IDs= matrix(dictionary$IDs_temp,,1)
> ORF= matrix(dictionary$ORF,,1)
> file_probeIDTu <-
+   "../data/MetabolData/probeIDTu.txt";
> probeIDTu <- read.table(file_probeIDTu, sep=" ", fill= TRUE, col.names=c("IDSn"))
> IDSnew <- matrix(probeIDTu$IDSn,,1)
```

The connectivity contains the evidence of connectivity between a gene and a transcription factor. `Connectivity2.txt` file depicts the relation between 6229 genes and 204 transcription factors. Load the connectivity file.

```
> file_dataChip <-
+   "../data/Connectivity2.txt";
> dataChip=as.matrix(read.table(file_dataChip))
```

From the annotation file we can find the gene names present in the connectivity information. Load the annotation file.

```
> file_annotation <-
+   "../data/annotations2.txt"
> probe_anno <- read.table(file_annotation, header = FALSE, sep = "\t", quote = "",
+                          dec = ".", col.names=c("prob", "anno"), na.strings = "NA",
+                          colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
+                          fill = TRUE, strip.white = FALSE, blank.lines.skip = FALSE,
+                          comment.char = "#", allowEscapes = FALSE, flush = FALSE)
> probeName2=matrix(probe_anno$prob,,1)
> annota=matrix(probe_anno$anno,,1)
```

The list of transcription factors present in the connectivity information can be found from the file

```
> file_transNames <-
+   "../data/Trans_Names2.txt"
```

```
> TransNames_tab <- read.table(file_transNames, header = FALSE, sep = "\t", quote = "",
+                              dec = ".", col.names=c("TN"), na.strings = "NA",
+                              colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
+                              fill = TRUE, strip.white = FALSE, blank.lines.skip = FALSE,
+                              comment.char = "#", allowEscapes = FALSE, flush = FALSE)
> TransNames = matrix(TransNames_tab$TN,,1)
```

In our experimental data set there are some genes for which the measurement of uncertainty is zero(i.e. vars[3,]). We would like to exclude those data.

```
> zeroValueRow = which(rowSums(vars)==0)
> data = data[-zeroValueRow, ]
> vars = vars[-zeroValueRow, ]
> probeName = ORF # Rename ORF
> probeName = matrix(probeName[-zeroValueRow, ],,1)
```

All the genes represent the connectivity information with the transcription factors might not present in the differentially expressed data set. We will like to exclude those genes. The following segment of code will exclude those genes. We will also exclude the redundant genes, their point expression data and uncertainty levels.

```
> noOfprobe=nrow(probeName)
> redundancy=matrix(mat.or.vec(noOfprobe,1),,1)
> redundancy[,]=1
> index=matrix(mat.or.vec(nrow(dataChip),1),,1)
> for (i in 1: nrow(probeName2)){
+         vec <- probeName==probeName2[i,1]
+         index[i]=colSums(vec)
+         if(index[i]>1){
+                 pippo=(which(vec))
+                 redundancy[pippo[2:length(pippo)]]=0
+                 }
+         }
> dataChip=dataChip[which(index!=0),]
> annota=annota[which(index!=0),]
> probeName2=probeName2[which(index!=0),]
> probeName2=matrix(probeName2,,1)
> probeName=probeName[which(redundancy!=0),]
> probeName=matrix(probeName,,1)
> data=data[which(redundancy!=0),]
> vars=vars[which(redundancy!=0),]
```

Again all the genes have the differentially expressed data may not have the connectivity information with the transcription factors. We will exclude those genes as well. We will also organize the genes connectivity information based on the differentially expressed genes index. So that in both data set (points expressions with uncertainty and connectivity information) have the common genes and aligned with the same index.

```
> preX=NULL
> annotation=NULL
> index=mat.or.vec(nrow(data),1)
> for (i in 1: nrow(data)){
+         index[i]=sum(probeName[i]==probeName2)
+         if (index[i]==1)
+                 preX=rbind(preX,dataChip[which(probeName[i]==probeName2),])
```

```
+                 annotation=rbind(annotation,annota[which(probeName[i]==probeName2)])
+ }
> data= data[which(index==1),]
> vars= vars[which(index==1),]
> probeName= probeName[which(index==1),]
> probeName=matrix(probeName,,1)
```

The connectivity matrix between genes and the transcription factors is a binary matrix. If there is an evidence that a gene is transcribed by a transcription factor or, a transcription factor is transcribing some other genes then the relation will be 1, the value will be 0 otherwise. We binarized the matrix by giving a value based on the suggestion of [Lee et al., 2002] when the associated p-value is less than 0.0001.

```
> X= mat.or.vec(nrow(preX),ncol(preX))
> I <- c(which(preX<1e-3))
> X[I] =1
```

In the experimental dataset there might have some genes which are not transcribing by any of the given transcription factors, again there might have some transcription factors which are not transcribing any of the present genes. We will exclude those genes and transcription factors.

```
> fakeX = rowSums(X)
> X=X[which(fakeX!=0),]
> annotation=annotation[which(fakeX!=0),]
> annotation = matrix(annotation,,1)
> effectX=colSums(X)
> TransNames=TransNames[which(effectX!=0),]
> TransNames=matrix(TransNames,,1)
> X=X[,which(effectX!=0)]; X[201:207,16:30]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
[1,]    0    0    0    0    0    1    0    0    0     0     0     0     0     0     0
[2,]    0    1    0    0    0    0    0    0    0     0     0     0     0     0     0
[3,]    0    0    0    0    0    1    0    0    0     0     0     0     1     0     0
[4,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0
[5,]    0    0    0    0    0    1    0    0    0     0     0     0     1     0     0
[6,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0
[7,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0
```

```
> data= data[which(fakeX!=0),]; data[1:5,1:7]
```

```
             V1          V2        V3        V4         V5          V6         V7
[1,]  7.6199683  7.00936333  6.327303  6.420305  6.4746593  6.42599518  5.9848784
[2,] 11.1160178 10.15776528 10.559681 10.625835 10.8329051 11.14180612 10.7627800
[3,]  5.6794099  5.94811239  5.656218  6.013510  6.6347741  6.82920197  6.4394373
[4,] -0.8461938 -0.09323972  1.174838  2.204736  0.7630712  0.05440951  0.9871222
[5,] 11.2509533 11.10515487 10.940308 10.467595 10.3794020  9.96586556  9.8538724
```

```
> vars= vars[which(fakeX!=0),]; vars[1:5,1:7]
```

```
             V1         V2         V3         V4         V5         V6        V7
[1,] 0.12919850 0.16833525 0.24385468 0.22452239 0.25321253 0.23656530 0.2989101
[2,] 0.10194004 0.14261692 0.12480590 0.12249678 0.11228873 0.09996799 0.1154624
[3,] 0.35203433 0.30378077 0.36894665 0.29941130 0.24717867 0.20727438 0.2476667
[4,] 2.14604703 1.89233701 1.47218035 1.15147407 1.67454729 1.86261881 1.5501621
[5,] 0.06955393 0.07214869 0.07750579 0.09116104 0.09549444 0.10993607 0.1157719
```

Define some variables-

```
> nGenes= nrow(data) # Number of genes will be present in our experiment
> npts= ncol(data) # Number of time points
> nTrans = ncol(X) # Number of TF will be present in our experiment
> muIn = array(0, dim <-c(nTrans,1));
> annotations = annotation # Both of the variavle contain the same data
> transNames = TransNames # Both of the variavle contain the same data
```

5 Reduce Variables

The connectivity matrix is a sparse matrix. Using `chipReduceVariables` we can find the same length integer vectors having the row indices, column indices, the values of the non-zero entries of the sparce matrix, and the number of effective genes

```
> R_C_V_nEffectGenes = chipReduceVariables(X);
> R = R_C_V_nEffectGenes[[1]]; R[1:10]

 [1] 289 458  17  27  35  47  59  75  98 102

> C = R_C_V_nEffectGenes[[2]]; C[1:10]

 [1] 1 1 2 2 2 2 2 2 2 2

> V = R_C_V_nEffectGenes[[3]]; V[1:10]

 [1] 1 1 1 1 1 1 1 1 1 1

> nEffectGenes = R_C_V_nEffectGenes[[4]]; nEffectGenes

[1] 580
```

Initialize some variables-

```
> diagonal = array(0.5, dim <- c(1,nTrans))
> precs_mat = array(1, dim <- c(nrow(vars),ncol(vars)))
> precs = precs_mat/(vars^2)
> beta=3;
> gamma=pi/4;
> params=matrix(c(beta,gamma,t(muIn),0.1*t(V), diagonal),1,)
```

6 Likelihood Optimization

We choose to optimize the likelihood using scaled conjugate gradient algorithm implement in `Netlab` [Nabney, 2002].

```
> options = array(0, dim <- c(1,18))
> options[1]=1; #display error values
> options[2]=0.0001 # measure of the absolute precision
> options[3]=0.0001  #objective function values between two successive steps to satisfy condit
> options[14]= 2000  # maximum number of iterations
> #options[17]=0.1
```

Optimize all the parameters using scaled conjugate gradient algorithm. This process is time consuming. The following line of command may take several hours to run. Here we can reduce the number of iteration to speed up the process and later we will load some optimized data. Figure. 1 shows the mean value of the transcription factors activity.

```
> options[14]= 3 # Number if iteration; only for trial purpose.
> #params = SCGoptimNoise(params, options, data, precs, X, nEffectGenes, R, C)
```

Load the optimized value-

```
> require("Matrix")
> source("../chipDynoLoadModules.R")
> load("../ResultsTu_Final.RData")
> V=params[(3+nTrans):(length(params)-nTrans)]
> preSigma <- as.matrix(sparseMatrix(R, C, x=V, dims = c(nEffectGenes,nTrans)))
> diagonal = params[(length(params)-nTrans+1) :(length(params))];
> Sigma = t(preSigma)%*%preSigma + diag(diagonal*diagonal)
> beta = params[1]
> gamma = params[2]
> mu = params[3:(2+nTrans)]
> #save.image("ResultsTu_test.RData") # To save the data
```

7 Computation of posterior expectations for a given gene and TF

chipDynoStatPostEst and chipDynoStatPostEstNoise computes posterior expectations of transcription factor activity for a transcription factor and a gene given the uncertainty of the expression level is absent or present respectively.

```
> #rm(list=ls())
> load("../ResultsTu_Final.RData")
> source("../chipDynoLoadModules.R")
> annotations = annotation
> transNames=TransNames
> transName= "ACE2"
> geneName="YHR143W"
> #geneName="KRE32"
>
> npts=ncol(data);
> nTrans=ncol(X);
> c = class(geneName)
> v= mat.or.vec(length(annotations),1)
> if (c == 'character'){
+   for (i in 1: nrow(annotations)) {
+                 v[i] <- geneName==annotations[i,1]
+                 }
+         x=data.matrix(X[which(v==1),])
+         data=t(data[which(v==1),])
+         precs = precs[which(v==1),]
+
+ } else if (c=='integer'){
+         x=data.matrix(X[geneName,])
```

```
> plot(mu, type='s', col='red', xlab= "Transcription Factors", ylab= "Mean Value")
```

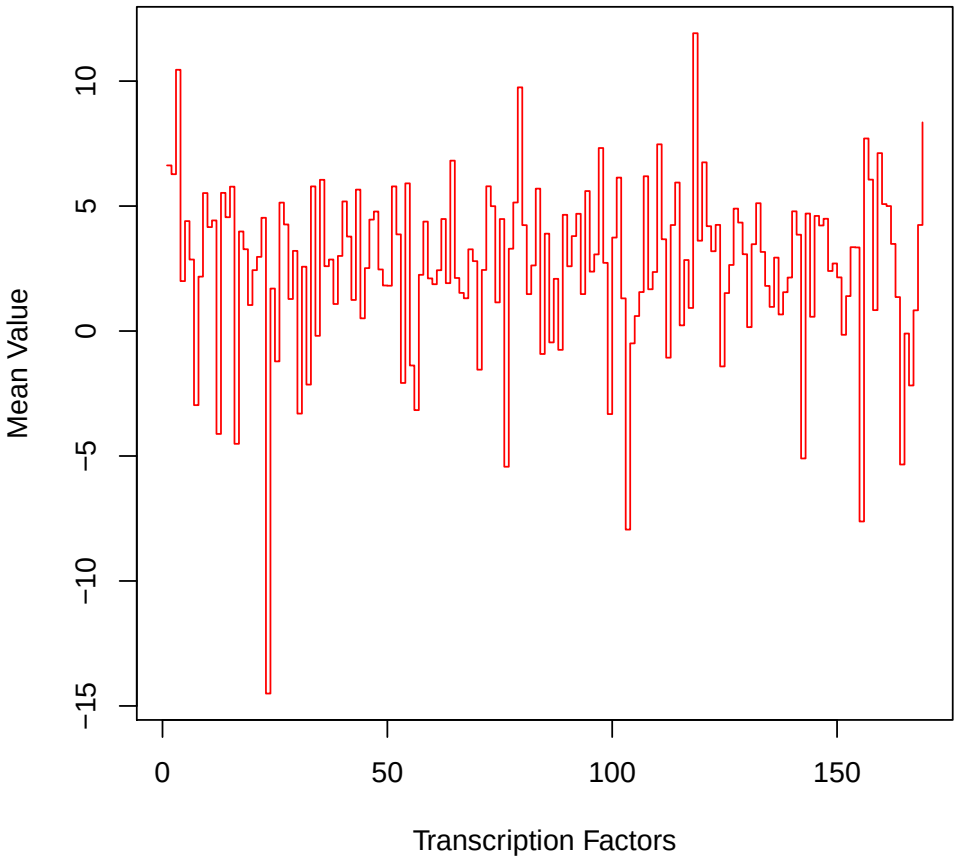


Figure 1: Posterior Mean of TFA

```
+         data=data[geneName,]
+         precs=precs[geneName,] ### t() ???
+
+ } else {
+         print('Error: Genes can be identified either by number or name')
+ }
> expectations=chipDynoStatPostEstNoise(data,x,Sigma,beta,precs,gamma,mu);
> expectations.b=expectations[[1]];
> expectations.tfError=expectations[[2]];
> expectations.tfErrorDiffs=expectations[[3]];
> index=which(transName==transNames);
> if (x[index,]== 0) {
+  print('Error: The gene selected is not a target of the transcription factor')
+ }
> tf=expectations.b[,index];
> ind=which(transName == transNames[which(x!=0)]);
> tfErrors=expectations.tfError[,ind];
> tfErrorsDiffs=expectations.tfErrorDiffs[,,ind];
```

```
> tf

 [1] 0.7626810 1.3371938 2.4517332 2.5973486 2.6137437 2.5888224 2.1749048 1.6951394
 [9] 2.1091986 1.0651819 1.0096594 0.9626544 0.8710457 1.5893280 2.6575006 2.5570126
[17] 2.3549824 2.6979573 2.2454622 1.8153979 1.3135126 1.2849328 1.4638310 1.0960801
[25] 1.0105326 1.9398684 2.4904246 2.7819267 2.5135438 2.3421242 2.1452290 2.2057685
[33] 1.6532838 1.3664687 1.2549145 0.9006209

> tfErrors

 [1] 2.773261 2.760529 2.758281 2.758231 2.758226 2.758233 2.758432 2.759117 2.758469
[10] 2.764042 2.762704 2.762955 2.770898 2.759345 2.758213 2.758245 2.758325 2.758204
[19] 2.758397 2.758859 2.760652 2.760853 2.759780 2.762497 2.768186 2.758657 2.758265
[28] 2.758191 2.758256 2.758334 2.758457 2.758394 2.759217 2.760254 2.760747 2.766046
```

8 Posterior expectations for a given TF : chipDynoExpectationsFastNoise

chipDynoExpectationsFast and chipDynoExpectationsFastNoise computes posterior expectations of transcription factor activity for a given transcription factor considering uncertainty of the expression level absent or present respectively. These functions first find all the genes gegulated by geven transcription factor and later the activity on different genes. The format of the function and arguments are-

```
> rm(list=ls())
> load("ResultsSpellman_Final.RData")
> #load("../ResultsTu_Final.RData")
> source("../chipDynoLoadModules.R")
> transNames = TransNames
> annotations = annotation
> nTrans=nrow(TransNames);
> lst=list();
> newX=array(0, dim <-c(dim(X)));
> newXVals=array(0, dim <-c(dim(X)));
> #i=2; name=TransNames[i,] # ACE2 for demSpellman
>
> name="ACE2"
> index=which(name == transNames);
> genesIn=which(X[,index]!=0);
> anno=annotations[which(X[,index]!=0)];
> nTargets=length(anno);
> npts=ncol(data);
> TF=array(0, dim=c(nTargets,npts));
> TFError=array(0, dim=c(nTargets,npts));
> TFErrorDiff=array(0, dim=c(npts,npts,nTargets));
> plotErrorBar <- function(y,SE, gnName){
+
+ add.error.bars <- function(x,y,SE,w,col=1){
+  x0 = x; y0 = (y-SE); x1 =x; y1 = (y+SE);
+  arrows(x0, y0, x1, y1, code=3,angle= 90,length=w,col=col);
+  }
+
+ x <- c(1:length(y));
+ plot(x,y, type = 'l', col='green4', las=1, xlab="time", ylab="TFA",
```

```
> M <- matrix(c(rep(1:4)), byrow=TRUE, nrow=2) # Choose the position by matrix setting!
> layout(M)
> k <- c(46,47,50,52)
> #for (i in 1:4){
> for (i in k){
+ #expectations = chipDynoExpectationsFast(data,X,Sigma,beta,gamma,mu,
+ #                    transNames, annotations, name, genesIn[i]);
+ expectations = chipDynoExpectationsFastNoise(data,X,Sigma,beta,precs,
+                     gamma,mu, transNames, annotations, name, genesIn[i]);
+ TF[i,] = expectations[[1]];
+ TFError[i,] = expectations[[2]];
+ gnName=annotation[genesIn[i]]
+ plotErrorBar(TF[i,],TFError[i,], gnName)
+ }
```

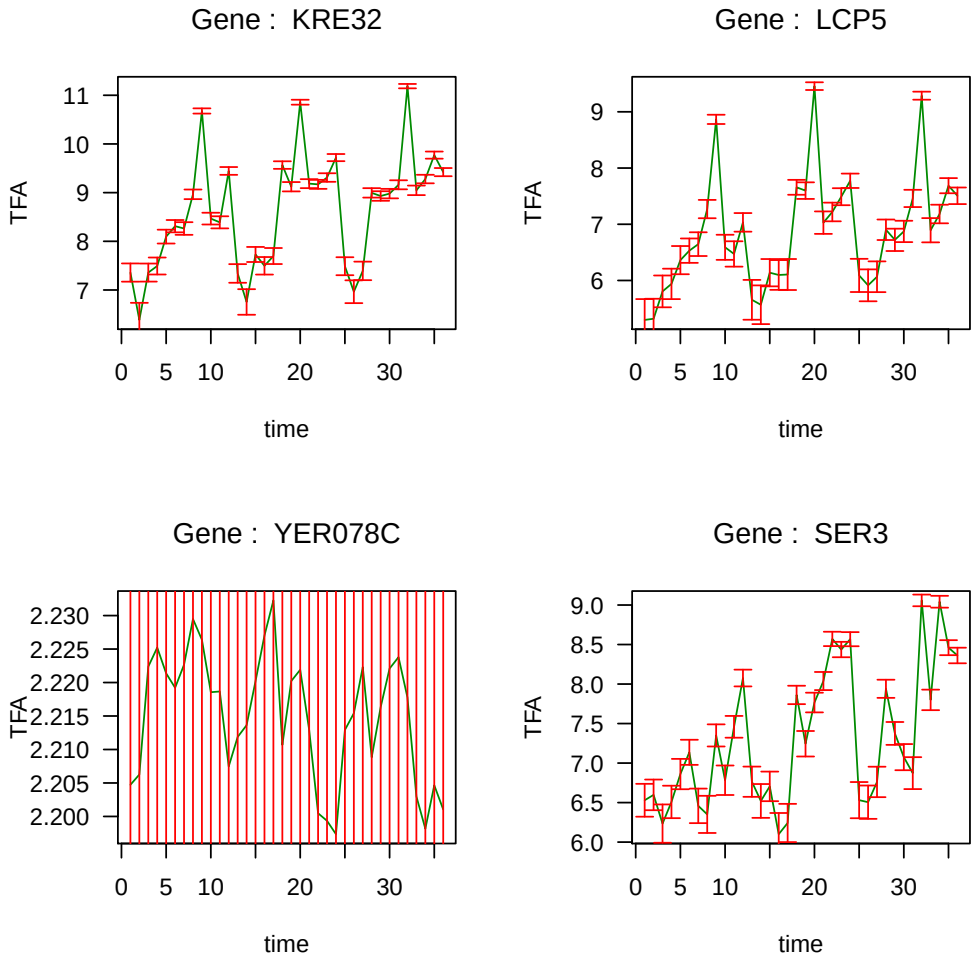


Figure 2: TFA Activities

```
+       main=bquote("Gene : " ~ .(gnName)));
+ add.error.bars(x,y,SE,0.05,col='red');
+ }
```

```
> M <- matrix(c(rep(1:4)), byrow=TRUE, nrow=2) # Choose the position by matrix setting!
> layout(M)
> k <- c(17,19, 28, 50)
> #for (i in 8:11){
> for (i in k){
+ expectations = chipDynoExpectationsFast(data,X,Sigma,beta,gamma,mu, transNames, annotations,
+ #expectations = chipDynoExpectationsFastNoise(data,X,Sigma,beta,precs, gamma,mu, transNames,
+ TF[i,] = expectations[[1]];
+ TFError[i,] = expectations[[2]];
+ gnName=annotation[genesIn[i]]
+
+ plotErrorBar(TF[i,],TFError[i,], gnName)
+ }
```

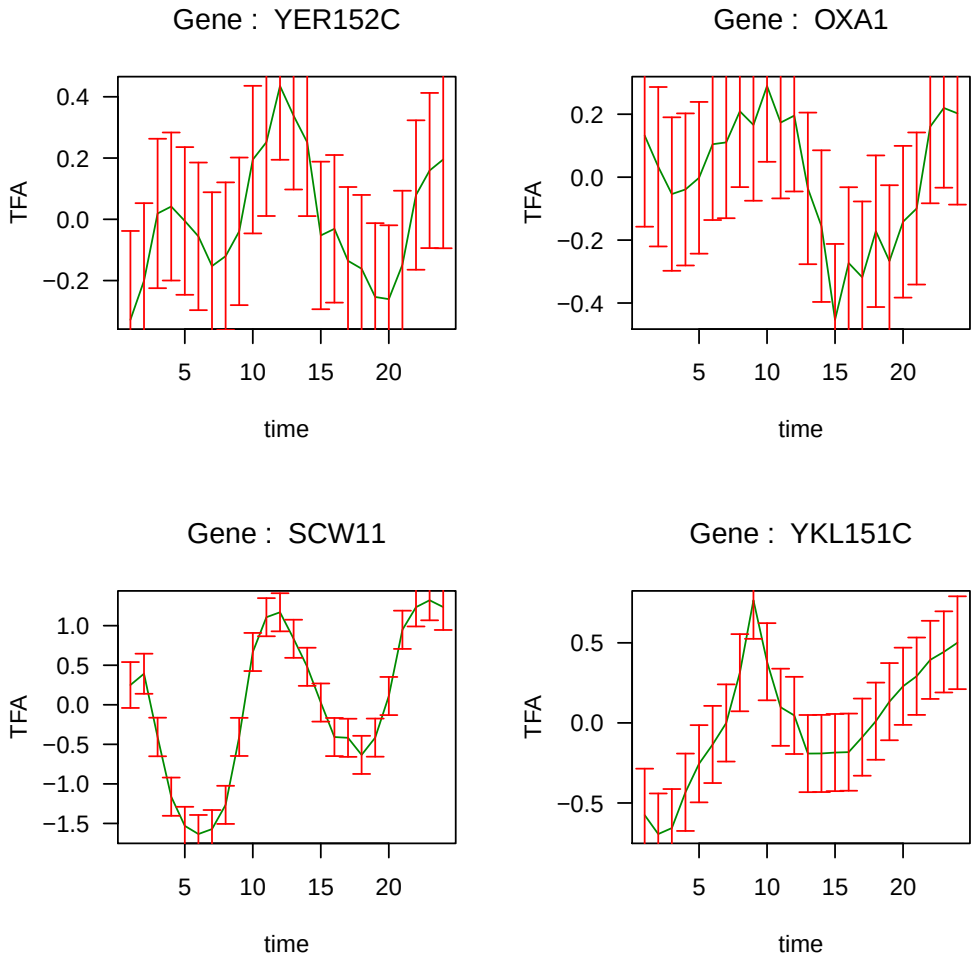


Figure 3: TFA Activities of ACE2

9 Maximum Transcription Factor Activity

chipDynoMaxDiff computes the maximum transcription factor activity of all the target or transcribed genes for a given transcription factor. For a specific target it also computes the variation over time.

```
> rm(list=ls())
> load("../ResultsTu_Final.RData")
```

```
> source("../chipDynoLoadModules.R")
> transNames = TransNames
> annotations = annotation
> nTrans=nrow(TransNames);
> lst=list();
> newX=array(0, dim <-c(dim(X)));
> newXVals=array(0, dim <-c(dim(X)));
> i=6; # TransNames[6,] is "AFT2"
> expectations =chipDynoTransFactNoise(data,X,Sigma,beta,precs, gamma, mu,
+                                      TransNames, annotation, TransNames[i,]);
> TF = expectations[[1]]
> TFError = expectations [[2]]
> TFErrorDiff = expectations [[3]]
> maxVars=chipDynoMaxDiff(TF,TFErrorDiff);
> #maxVars
```

10 TFAs with error bars :chipDynoTransFactNoise

chipDynoTransFactNoise provides transcription factor activities with error margin for a given transcription factor. For an example we are interested to find the activity of transcription factor ACE2. Using chipDynoTransFactNoise we can find it by following way-

```
> rm(list=ls())
> load("../ResultsTu_Final.RData")
> #load("../ResultsTu.RData")
> source("../chipDynoLoadModules.R")
> i=4; TransNames[i,]

[1] "ACE2"

> expectations =chipDynoTransFactNoise(data,X,Sigma,beta, precs, gamma, mu,
+                                      TransNames, annotation, TransNames[i,]);
> TransNames[i,] # Name of the transcription factor

[1] "ACE2"

> expectations[[1]][1:5,1:5] # Transcription facot activity on different genes

            [,1]      [,2]       [,3]       [,4]       [,5]
[1,]  1.11519852 1.4685515 1.92696131 2.06662345 2.09650500
[2,]  2.93071966 2.7465487 2.57092062 2.80903684 2.90590250
[3,] -0.03532873 0.1613834 0.05887395 0.06218098 0.07963148
[4,]  0.75657027 0.8134580 0.74290215 0.74045359 0.74227924
[5,]  1.97300550 1.9733308 1.96459567 1.96068312 1.96982907

> expectations[[2]][1:5,1:5] # Transcription facot activity error on different genes

         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 2.825080 2.824669 2.824563 2.824552 2.824549
[2,] 2.537083 2.537217 2.537447 2.537162 2.537087
[3,] 3.442080 3.441786 3.441868 3.441853 3.441868
[4,] 3.441750 3.441747 3.441751 3.441751 3.441751
[5,] 3.564012 3.564012 3.564012 3.564012 3.564012

> #expectations[[2]]
```

11 Significantly varying TFs :chipDynoActTransFactNoise

chipDynoActTransFact and chipDynoActTransFactNoise can find out transcription factor activity for all the transcription factors without or with uncertainty of expression level respectively

```
> rm(list=ls())
> load("../ResultsTu_Final.RData")
> source("../chipDynoLoadModules.R")
> nTrans=nrow(TransNames);
> lst=list();
> newX=array(0, dim <-c(dim(X)));
> newXVals=array(0, dim <-c(dim(X)));
> i=1
> expectations =chipDynoTransFactNoise(data,X,Sigma,beta,precs, gamma, mu,
+                                      TransNames, annotation, TransNames[i,]);
> expectations[[1]][1:5,1:5] # Transcription Factor activity

         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 3.808254 3.819266 3.580342 3.530811 3.639116
[2,] 7.992771 9.052932 8.876395 8.822699 9.228027
[3,] 6.278216 6.262015 5.851471 5.898305 6.425768
[4,] 9.367163 8.709554 7.501238 7.325142 7.652420
[5,] 5.749316 5.776974 5.347983 5.821693 6.240452

> expectations[[2]][1:5,1:5] # Transcription Factor activity error

           [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 3.28911886 3.2889320 3.2907785 3.2907267 3.2914427
[2,] 0.19198452 0.1224492 0.1338201 0.1347550 0.1192488
[3,] 0.23749561 0.2267321 0.2941456 0.2797092 0.2359230
[4,] 0.09053301 0.1132712 0.1933534 0.2011850 0.1860413
[5,] 3.69168251 3.6916071 3.6926496 3.6915638 3.6912072

> expectations[[3]][1:5,1:5,1] # Transcription Factor activity error difference

          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0000000 0.4844291 0.6761753 0.8177611 0.9372513
[2,] 0.4844291 1.0000000 0.4897352 0.6755841 0.8188247
[3,] 0.6761753 0.4897352 1.0000000 0.4951315 0.6817655
[4,] 0.8177611 0.6755841 0.4951315 1.0000000 0.4971175
[5,] 0.9372513 0.8188247 0.6817655 0.4971175 1.0000000
```

12 Given Gene: lists activators in decreasing order

For a given gene we can find out the list of activators using the function chipDynoGeneActNoise. It will list all the activators in decreasing order with transcription factor activity and activity error.

```
> rm(list=ls())
> load("../ResultsTu_Final.RData")
> source("../chipDynoLoadModules.R")
> geneName="YHR143W";
> #geneName="AGA1";
```

```
> transNames=TransNames
> activators = chipDynoGeneActNoise(data, X, Sigma, beta, precs, gamma,mu,
+                     transNames, annotation, geneName)
> activators[[1]] # List of activators

[1] "ACE2" "FKH2" "FKH1" "MBP1"

> activators[[2]] # Maximum transcription factor activity

[1] 0.7841517 0.7258846 0.4750492 0.1858749

> activators[[3]] # Maximum transcription factor activity error

[1] 2.758191 3.198300 2.983093 3.743103
```

13 Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
> toLatex(sessionInfo())
```

- R version 3.0.2 (2013-09-25), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_GB.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB.UTF-8`, `LC_COLLATE=en_GB.UTF-8`, `LC_MONETARY=en_GB.UTF-8`, `LC_MESSAGES=en_GB.UTF-8`, `LC_PAPER=en_GB.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_GB.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: Matrix 1.1-2
- Loaded via a namespace (and not attached): BiocStyle 0.0.19, grid 3.0.2, lattice 0.20-24, tools 3.0.2

References

[Lee et al., 2002] Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. T., Thompson, C. M., Simon, I., Zeitlinger, J., Jennings, E. G., Murray, H. L., Gordon, D. B., Ren, B., Wyrick, J. J., Tagne, J.-B., Volkert, T. L., Fraenkel, E., Gifford, D. K., and Young, R. A. (2002). Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(14):799–804.

[Nabney, 2002] Nabney, I. T. (2002). Netlab: Algorithms for pattern recognition. *Springer*. London.

[Pearson et al., 2009] Pearson, R., Liu, X., Sanguinetti, G., Milo, M., Lawrence, N., and Rattray, M. (2009). puma: a bioconductor package for propagating uncertainty in microarray analysis. *BMC Bioinformatics*. 10:211.