2010-12-10

# Gene Network Inference and Expression Prediction Using Recurrent Neural Networks and Evolutionary Algorithms

Heather Y. Chan
*Brigham Young University - Provo*

# Gene Network Inference and Expression Prediction Using Recurrent Neural Networks and Evolutionary Algorithms

Heather Y. Chan

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Quinn O. Snell, Chair
Mark J. Clement
Jay A. McCarthy

Department of Computer Science

Brigham Young University

April 2011

## ABSTRACT

Gene Network Inference and Expression Prediction Using Recurrent Neural Networks and Evolutionary Algorithms

Heather Y. Chan

Department of Computer Science

Master of Science

We demonstrate the success of recurrent neural networks in gene network inference and expression prediction using a hybrid of particle swarm optimization and differential evolution to overcome the classic obstacle of local minima in training recurrent neural networks. We also provide an improved validation framework for the evaluation of genetic network modeling systems that will result in better generalization and long-term prediction capability. Success in the modeling of gene regulation and prediction of gene expression will lead to more rapid discovery and development of therapeutic medicine, earlier diagnosis and treatment of adverse conditions, and vast advancements in life science research.

Keywords: genetic network modeling, gene network inference, gene expression prediction, recurrent networks, evolutionary algorithms, time series prediction

ACKNOWLEDGMENTS

I would like to extend heartfelt thanks to my advisor, Dr. Quinn Snell, for his continual help and advice in what has been a long journey. I also would like to thank him, Dr. Mark Clement and Dr. Jay McCarthy for their repeated readiness to put in hard work on short notice to accommodate my unusual circumstances.

My family and friends have been a wonderful network of support as I have worked to complete this research long-distance, away from my peers and mentors. I would like to acknowledge and express my appreciation for their love and support over the last two years.

Finally, I would like to thank Tony Lenzo, STAR LAB System Administrator at the University of Rochester Political Science Department, for allowing me to use lab resources to successfully complete my research.

## Contents

## List of Figures

# List of Tables

# Chapter 1

# Introduction

Gene regulatory network modeling is an important area of computational biology because it increases understanding of biological systems in terms of both function and response to external stimuli. Gene expression data has become abundant through the recent development of DNA microarray technology, facilitating research in this area. However, the complexity of gene interactions makes it difficult to determine relationships through static measurements alone; computational models capable of processing time series expression data thus prove to be essential tools in uncovering meaning in vast amounts of data.

## 1.1 Background

Genes are sections of DNA that encode information for protein production. Each gene consists of a DNA nucleotide sequence that is transcribed by RNA polymerase into an mRNA nucleotide sequence. The mRNA is then translated by ribosomes into an amino acid sequence, which folds into a protein. Proteins play a central role in living organisms by participating in every cellular process. Some catalyze biochemical reactions that are essential for metabolism; some have structural and mechanical functions; some are responsible for cell signaling and immune responses.

Expressed proteins may regulate other genes through activation or inhibition. In activation, the presence of the regulating compound causes increased expression of the regulated gene. In inhibition, the presence of the regulating compound causes decreased expression of the regulated gene. The coexistence of numerous positive and negative regulatory relationships among a set of genes results in a complex gene regulatory network that governs how the cell reacts to various events. Knowledge of this type of network is essential in order to develop drugs to counter adverse effects.

### Drug discovery and development

Despite rapidly advancing technology and continually growing understanding of biological systems, the process of drug discovery and development for therapeutic medicine remains a time-consuming, expensive undertaking. According to the website for Pharmaceutical Product Development, Inc., the discovery and development of a single new drug takes an average of 10 to 15 years and can cost anywhere from $800 million to more than $1 billion [25]. Drug candidates can sometimes be identified using knowledge about the pharmacological properties of known substances, but many effective drug compounds have only been discovered through random collection and screening of data, which is both costly and inefficient. Since drug candidate identification can be quite difficult and is often purely accidental, enhanced methods to automate and expedite drug screening are absolutely essential to achieve cost reductions and enable higher throughput in the drug development process.

### Microarray technology and *in silico* methods

With the advent of microarray technology, researchers are able to collect many times more genetic data than ever before over a relatively short period of time. A single microarray chip contains thousands of wells of genetic material that can be simultaneously tested and measured using fluorescent tagging, image capture, and image processing algorithms. However, the process of constructing and evaluating hypotheses to describe these large amounts of data is an extremely difficult problem that continues to perplex researchers.

Due to the high dimensionality of microarray data and relative sparsity of underlying biological relationships that govern it, there are infinitely many hypotheses that can be constructed to fit the data. Exhaustive exploration and isolated testing of all possible genetic

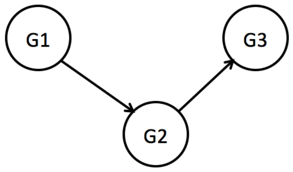


Figure 1.1: Simple gene regulatory network represented as a directed graph

relationships in a lab setting is prohibitively time- and cost-intensive as it would involve thousands of dead-end experiments and excessive waste of costly genetic materials.

In contrast to experiments conducted *in vivo* and *in vitro*, methods designated as *in silico* are run entirely in computer simulation and are intended to take advantage of the abundant data that has already been collected. *In silico* methods seek to find patterns within existing data and do not have the same demands on time and resources as their "wet lab" counterparts, nor do they carry the latter's risks of contamination, erroneous errors and operator-related variability. These benefits highlight *in silico* methods as an important supplementary technique in genetic regulatory network discovery that is significantly less expensive and time-consuming.

## The challenges of genetic network modeling

A gene regulatory network can be represented as a directed graph, where nodes represent genes and arcs represent regulatory influences between genes. For example, Fig. 1.1 depicts a gene regulatory network of three genes G1, G2, and G3, where G1 influences G2 and G2 influences G3. Depending on the desired representational power, this graph can be mathematically expressed by a set of Boolean functions, as a joint Bayesian probability distribution, by a set of linear differential equations, stochastic equations, or rule-based formalisms, among several other mathematical formalisms that have been employed in this context [9]. The choice of representation determines whether the model can describe continuous expression levels, whether it can represent linear and nonlinear dynamics, and whether it can account for directed and cyclic relationships.

As in all types of modeling, significant tradeoffs exist among representational power, scalability, and interpretability. It is paramount that the chosen model have the expressive capability necessary to describe real biological networks, else no amount of training and datasets can possibly produce an adequate hypothesis. However, for relevancy to real-world problems involving hundreds to thousands of genes, the model must also be highly scalable and store information in an easily interpretable format. In addition, the hypotheses that it produces must be biologically plausible.

Another challenge that arises in genetic network modeling is that there are infinitely many hypotheses that can be constructed to fit the data. Hence, one of the foremost priorities in the design of an inference system is the choice of reliable evaluation criteria to assess the fitness of a solution and place many equally well-fitting hypotheses into a ranked order. For example, one way to distinguish the merit of a set of solutions is to compare their consistency with existing knowledge of the biological network. Another is to take advantage of the fact that biological networks tend to be relatively sparse, making it likely that sparser networks are more plausible than dense networks.

### Time series data and gene expression prediction

The end goal of gene regulatory network inference is an understanding of the underlying interactions of genes, and thus the ability to determine the behavior of the network when changes occur in the concentration levels or transcription rates of the various gene products. For example, once the network is reconstructed, one can hypothesize what would be likely to occur if a new compound was introduced into the system.

If time series data is available, then gene expression prediction can be used as another way to evaluate how good a candidate solution is. For example, if the chosen model can describe the dynamics of genetic expression, a novel set of initial conditions should produce the correct trajectories and arrive at the correct steady state. On the other hand, network inference systems that do not take dynamics into account cannot take advantage of temporal clues and may fall into the trap of overfitting to noise and defects in the training data. Overfitting precludes generalization and undermines confidence in the inferred relationships.

With few clues to go on for network inference other than existing biological knowledge and the tendency for biological networks to be relatively sparse, we assert that evaluation of gene expression prediction is a significant and valuable method to further narrow the hypothesis space and increase the likelihood that inferred models will approach truth.

## 1.2 Problem Description

Given a set of $N$ genes and time series data reflecting changes in their expression levels over time, we infer a model that accurately describes and reflect the dynamics of the regulatory network formed by those genes. In order to do this, our model learns a temporal relationship between the current expression levels of each gene and the expression levels of each gene at a future point in time. In addition to modeling the temporal behavior of the genes, we are also able to extract meaningful information from our model in the form of directed causal relationships between genes, such as activation and inhibition.

The existence of infinite possible hypotheses consistent with a particular set of gene expression data demands network inference methods that can sift through and keep only the most biologically plausible hypotheses. We have thus devised a method to incorporate existing biological knowledge into our model as a means of restricting the hypothesis space to more biologically valid solutions. We also use evaluation of gene expression prediction to further eliminate hypotheses that fit the data but cannot generalize to novel conditions.

Finally, our model is able to make accurate and usable predictions of gene expression time series data. Although most current network inference models are not capable of extended time series predictions without severely adverse influences of noise and drift, we consider it a highly important to be able to predict longer-term trends. This type of prediction will allow for *in silico* perturbation tests on genetic networks, where new initial conditions or modifications to gene dependencies will result in new behavior over time. These types of tests will be useful in the study of reactions to new compounds and hence crucial to the facilitation of *in silico* drug pre-screening.

## 1.3 Thesis Statement

Recurrent neural networks are capable of modeling genetic network dynamics in real biological systems with better generalization and long-term prediction accuracy than other state-of-the-art methods. They achieve their best performance and scalability when trained with genetic algorithms to avoid the pitfalls of local minima and complex derivatives in gradient descent algorithms. Long-term gene expression prediction accuracy is crucially indicative of the plausibility of an inferred genetic network model and can be measured using magnitude squared coherence.

# Chapter 2

# Related Work

In order to illustrate the rationale behind our choice of model, we present an overview of numerous model representations that have been employed by other researchers in this field along with their strengths and weaknesses. We also discuss the particular computational challenges of genetic network modeling as well as recent attempts to address them.

## 2.1 Representations of Genetic Networks

The earliest methods for gene regulatory network inference consisted of gene clustering based on similarity of expression profiles. This approach was motivated by the hypothesis that genes showing similar expression profiles are likely to be co-regulated. However, clustering methods are not able to describe causal relationships between gene clusters, limiting their usefulness in this context. Several mathematical formalisms have since been employed to describe and represent gene regulatory networks from an interactional standpoint.

### Boolean networks

Boolean networks describe regulatory network structure by representing genes as nodes in a network and modeling the state of each node as either ON or OFF. The current state of each node is described by a Boolean function of the outputs of the other nodes. This discrete representation facilitates computation and inference of relationships between genes and has been widely employed by many researchers. However, the representation of gene expression as a binary state is a somewhat crude assumption, and a Boolean representation of gene expression may not possess the full representative power required for the inference problem. Probabilistic Boolean networks have been employed to address this by probabilistically modeling network dynamics while retaining the rule-based advantages of Boolean networks [27]. However, many have turned to more complex models in order to capture the complex behavior of gene network dynamics.

### Bayesian networks

Bayesian approaches have been used to infer gene interactions using data from multiple expression measurements [13] and to construct probabilistic gene regulatory networks while focusing on network connectivity [42]. Bayesian networks effectively deal with noise, incompleteness, and the stochastic nature of gene expression, and their graph representation is intuitive. However, Bayesian networks, like Boolean networks, generally do not consider dynamics or incorporate temporal information. Dynamic Bayesian networks attempt to address this through unrolling of time steps in order to take advantage of sequential data points. An influence scoring method for dynamic Bayesian networks has been shown to improve results when combined with limited observational data [41].

### Differential equations

Differential equations, in contrast to the other methods mentioned so far, are quite capable of capturing and modeling complex network dynamics. For small networks, differential equations have been used to model gene inhibition and activation according to engineering paradigms from control theory. However, it remains unknown whether genetic interactions bear any resemblance to manmade equations. Also, differential equations, which must be formulated by hand, quickly become very complex and are hard to construct for large networks (upwards of thousands of genes), making them unwieldy for larger applications.

## Probabilistic and stochastic models

Robustness and scalability are essential features of a usable model. Shmulevich et al. and Tian and Burrage address robustness by introducing probabilistic and stochastic elements into their systems to account for uncertainty [27, 31]. Bock and Gough use learned decision functions to predict interactions for similar species, which may prove useful for generalization of interactions based on species similarity and therefore scalability [4]. Wang et al. (2006) explore the possibility of using parallel processing to reduce computational load and increase scalability [35].

## Neural networks

Neural networks are one of the most promising methods for gene regulatory network inference; they are able to model complex network dynamics and can be automatically trained and decomposed into smaller units to take advantage of parallel processing speed boosts. They have both the necessary representational power that differential equations possess and a modest degree of scalability. Neural networks have been used to model the dynamics of gene expression, some methods modeling transcription and translation independently [32]. Stochastic neural networks have also been used to describe intermediate regulation for large-scale gene networks, using fluctuation variables as a way to study robustness and stability properties through simulation [31]. The largest disadvantages of neural networks are their slow training time and the difficulty of interpreting the learned weights into useful biological information. However, recent developments have begun to address these issues with some success, most notably the use of evolutionary algorithms to reduce training time [39], the development of statistical methods to extract relationships from network weights [24], and the incorporation of fuzzy rules into neural networks to improve readability [20].

## Recurrent neural networks

Recurrent neural networks have the complex representational power and decomposability of standard feedforward neural networks, but they are additionally able to represent temporal dependencies and cyclic relationships such as self-activation and self-inhibition. Until recently, recurrent neural networks were not a competitive option for genetic network modeling due to the difficulty of training them. While feedforward neural networks are commonly trained using backpropagation, a gradient-descent learning algorithm that iteratively refines the weights on each node in the network to approximate the behavior of the training data, standard backpropagation cannot be applied to recurrent networks because of the presence of cycles in the network. There is version of backpropagation for recurrent networks called Backpropagation Through Time (BPTT) and described and outlined by Paul Werbos (1990), but BPTT has the tendency to get stuck in local minima and has not proved effective in this application. However, new training methods have emerged that make it possible to apply recurrent neural networks to gene network modeling.

Several researchers have recently employed a recurrent neural network approach for gene network modeling. Dai reports success in modeling the synthetic oscillatory network of Escherichia coli cells using an echo state network [7]. Hu et al. use a recurrent neural network to model dynamics of gene networks and learn their parameters, defining positive and negative regulations by a weight matrix and allowing distinct decaying time constants for each gene [16]. Lee and Yang establish a cluster-based inference method for recurrent neural networks and employ feature extraction to deal with scalability, recursively clustering genes into smaller network components [18]. Xu et al. train their recurrent neural network with hybrid differential evolution and particle swarm optimization to infer genetic regulatory networks from time series data [39].

We find that recurrent neural networks are overwhelmingly the approach with the most potential for working with and taking advantage of the temporal information afforded by using time series data. They possess rich representational power that can describe the dynamics of both directed and cyclic relationships, and they are not constrained or biased by predefined rule structures found in neurofuzzy networks. Evolutionary algorithms rectify the problems of slow training and entrapment in local minima, and omission of hidden nodes allows network weights to be directly read as causal relationships between nodes.

## 2.2 Current Issues in Genetic Network Modeling

### Collection and interpretation of time series expression data

The study and analysis of time series expression data is an extremely important area of research for computational biology, as temporal information serves as an important additional clue for the inference of causal relationships. Bar-Joseph presented a comprehensive review of research in time series expression data analysis, outlining the computational challenges in four analysis levels [1]. He defines these levels as experimental design, data analysis, pattern recognition, and networks.

First, experimental design introduces the problem of determining sample rates, where under-sampled results might be an incorrect representation of the activity and miss key events, and over-sampling is computationally time consuming. Ideally, sampling rates should be related to transcription and degradation rates of mRNA.

Second, at the data analysis level, the goal is to construct a continuous representation of all genes over the course of the entire experiment. However, microarray data is very noisy and there are few replicates, creating barriers to reliable interpolation; in addition, different organisms may undergo similar processes at very different rates, making it difficult to combine datasets across experiments.

Third, pattern recognition deals with effective data organization and visualization, and can benefit from known dependencies in time series data. An important difference when dealing specifically with time series expression datasets is that the relationships between clusters are as important as clusters themselves.

Finally, the networks level is where descriptive and predictive models are built to explain the behavior of genes. The largest difficulty at this level is the limitation of expression data, necessitating the addition of biological data to constrain the number of different hypotheses that are consistent with the data. Another challenge is obtaining perturbation data, which for regulatory networks consists of gene knockouts under different conditions.

## Incorporation of temporal information in model inference

Several recent methods have attempted to utilize the temporal information available only in time series data. Haverty et al. use data from stimuli responses to identify transcription factors that regulate gene expression factors, and can predict which genes are regulated by each transcription factor [15]. Chen et al. propose a differential equation approach modeling both feedback loops from translation to transcription and degradation of proteins and mRNAs; their results suggest that only a minor set of accurate temporal data is required for model construction [6]. Barker et al. proposed a method to incorporate temporal information to achieve better predictions on causal network connectivity [2]. Their method calculates potential influence vectors for each gene based on probability ratios; vectors are combined and competed against each other to reach a final influence vector for each gene.

The use of recurrence is highly appropriate for studies involving time series data because of its ability to represent temporal and sequential dependencies. A study using recurrent networks for financial time series prediction shows more accurate predictions compared to traditional statistical and feedforward approaches [40]. Also, results from a comparative study on backpropagation for feedforward networks versus backpropagation through time for recurrent networks show that backpropagation through time has superior performance, being generally more robust to noise and having a faster convergence rate [30]. Werbos, the first to outline the algorithm for backpropagation through time, describes applications to pattern recognition for dynamic systems, systems identification, and control [36]. Giles et al. apply recurrent neural networks to financial forecasting, converting data into a symbolic representation and doing grammatical inference with some success in predictability [14].

The positive results of these studies on recurrent neural networks with respect to time series data demonstrates their relevancy to the problem of genetic network inference and expression prediction. The improvement gained by using temporal information is a significant benefit to systems that seek to describe and predict the behavior of genetic networks.

## Gene expression prediction

Ronen et al. were among the first to develop a system for predicting gene expression values [26]. They demonstrated the usefulness of kinetic parameter estimation in predicting trajectories for several genes with the knowledge of only one gene's trajectory. Maraziotis et al. also achieved considerable success in gene expression prediction for a single gene given all expression levels at the current timestep and the expression levels for all other genes at the next timestep [21]. They employed a recurrent neurofuzzy network that learned fuzzy rules to determine the behavior of the gene under study. However, we take this further by using only information given in the current timestep to predict the next. The significance of this achievement is the ability to predict long-term trends without relying on knowledge of other gene products. We view this capability as crucial for *in silico* drug pre-screening.

Smith et al. demonstrated that neural networks with hidden layers and simple Elman networks outclass other types of neural networks in terms of prediction accuracy and ability to settle to steady states [28]. Drawing from their conclusions, we compare our model with these two types of networks in order to evaluate our model's prediction accuracy in both the short term and the long term.

## Incorporation of *a priori* biological knowledge

The ability to account for existing biological knowledge is important because it constrains the hypothesis space as well as ensuring that results are consistent with biological studies. Wang et al. use linear programming to incorporate known biological properties [35], and Barker et al. allow for the incorporation of known gene relationships prior to training their system [2]. In addition to being consistent with biological studies, it is important in terms of biological plausibility to achieve sparsity in network models, as sparsity is a strong characteristic of biological systems. Wang et al. address this by maintaining sparsity throughout the search for a network structure that is the most consistent with all data [35]. Bhadra et al. do this similarly by formulating the structure estimation problem as a sparse linear regression problem [3]. For neural networks without hidden layers, a possible approach to incorporate existing knowledge is to first infer the connectivity of a network using a separate method, and then hold all excluded weights to zero. This final method is the one we employ.

### Dealing with complexity and interpretability

Xu et al. praise the ability of recurrent neural networks to interpret complex temporal behavior and point out similarities between recurrent networks and gene networks [37, 38, 39]. They also suggest methods for dimensionality reduction, including clustering, interpolation, adding noisy duplicates, thresholding, and strategies for network training. Some of their most recent work demonstrates that time demands in training recurrent neural networks can be greatly reduced by using a hybrid of differential evolution and particle swarm optimization to train the weights of the networks. This advancement addresses the problem of slow training in neural networks. Xu et al. also addressed the problem of interpretability by incorporating the statistical method used by Perrin et al. to extract relationships from the weights of the network [24]. Maraziotis et al. took another approach, introducing fuzzy rules to produce a neurofuzzy network retaining the computational power of recurrent networks, yet yielding interpretable rules [20]. However, due to the bias introduced by fuzzy rule structure, we prefer the representational power of classic recurrent neural networks.

# Chapter 3

# Methodology

## 3.1 Model

We employ artificial recurrent neural networks to model gene regulatory networks with the specific goal of being able to predict trends in gene expression. Artificial neural networks are mathematical tools for nonlinear statistical data modeling. They can model complex, nonlinear functions, and their modular design allows them to be used in a parallel processing environment. Artificial neural networks have been widely employed in solving biological problems such as predicting the secondary structure of proteins. Recurrent neural networks (RNNs) are a class of neural networks that contain directed cycles between units, allowing for dynamic temporal behavior (see Fig. 3.1).

Our model contains one input node and one output node for every gene, and the network is fully connected, i.e. every input node is connected to every output node. In

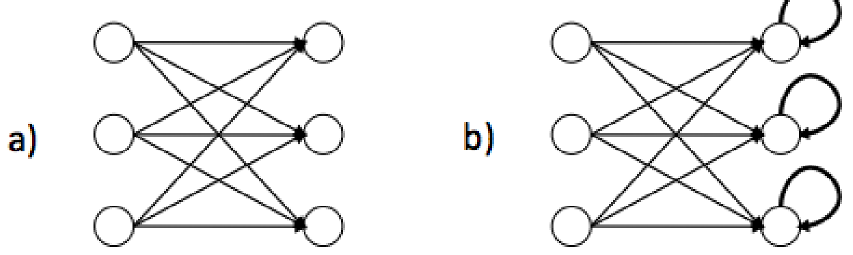


Figure 3.1: Basic feedforward neural network (a) vs. recurrent neural network (b)

addition, every output node contains a cycle back to itself. Fig. 3.1(b) is an example of the network structure for a 3-gene dataset.

Equation (3.1) is the mathematical formula that we use to calculate the output of each node and is used by Xu et al. [39].

$$e_i(t + \Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t), \tag{3.1}$$

where $e_i$ is the gene expression level for the $i$th gene ($1 \leq i \leq N$, $N$ is the number of genes in the system), $f(\cdot)$ is a sigmoid function $f(z) = 1/(1 + e^{-z})$), $w_{ij}$ represents the effect of the $j$th gene on the $i$th gene ($1 \leq i, j \leq N$), $\tau$ is the time constant (controlling time decay), and $\beta$ is the bias term. Negative values of $w_{ij}$ indicate inhibition of the $j$th gene on the $i$th gene, and positive values indicate activation. Zero values indicate no influence.

The quantity $f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right)$ in Equation 3.1 corresponds to the form $f(weight\_vector \cdot x + bias)$ that a perceptron, the basic unit of the neural network, uses to map its input $x$ to an output value; in Fig. 3.2(a), this quantity describes the connections from the input layer $e(t)$ to the output layer $e(t+\Delta t)$. If there were no recurrent connections in the network (as in Fig. 3.1(a)), the equation for the network outputs would simply be $e_i(t+\Delta t) = f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right)$. Equation 3.1 augments this nonrecurrent model with the previous output value $e_i(t)$ to incorporate temporal (recurrent) information; in Fig. 3.2(a), this describes the cyclic connections in the output layer. The coefficients $\frac{\Delta t}{\tau_i}$ and $\left(1 - \frac{\Delta t}{\tau_i}\right)$ are used to weight the nonrecurrent and recurrent portions of the equation.

In a study of the effect of hidden nodes in recurrent networks, Equations 3.2 and 3.3 incorporate a layer of nonrecurrent hidden nodes into the Xu model, mapping inputs $e(t)$ into outputs $h(t)$ and using the mapped values in the Xu equation instead of directly using the input values. The intention here is to preprocess the inputs and map them into another space before feeding them into the Xu model. These modifications are shown in Fig. 3.2.

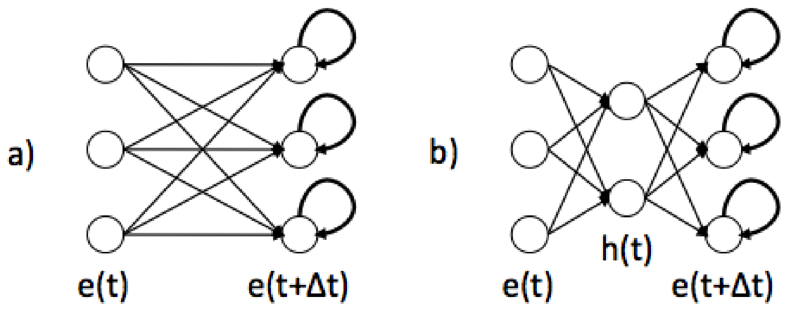


Figure 3.2: Recurrent neural network without hidden layer (a) vs. with hidden layer (b)

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{H} w_{ij} h_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t), \tag{3.2}$$

$$h_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{N} v_{ij} e_j(t) + \gamma_i\right), \tag{3.3}$$

where $h_i$ is the output of the $i$th hidden node ($1 \leq i \leq H$, $H$ is the number of hidden nodes in the network), $w_{ij}$ represents the effect of the $j$th hidden node on the $i$th gene ($1 \leq i \leq N, 1 \leq j \leq H$), $v_{ij}$ represents the effect of the $j$th gene on the $i$th hidden node ($1 \leq i \leq H, 1 \leq j \leq N$), and $\gamma$ is the bias term for the hidden node.

Table 3.1: Comparison of values of $H$ hidden nodes for SOS dataset

| | H=0 | H=5 | H=10 | H=15 | H=20 |
|---|---|---|---|---|---|
| Training MSE | 0.00183 | 0.00193 | 0.00193 | 0.00309 | 0.00192 |
| Test MSE | 0.00341 | 0.00323 | 0.00431 | 0.00350 | 0.00851 |

We have experimented with hidden layers of nodes between the input and output layers, but have chosen not to use hidden layers for two reasons. First, it is much easier to extract causal relationships from a simple 2-layer network than one incorporating several combinations of weights from different nodes. Second, the performance of the recurrent network does not show noticeable improvement through the addition of hidden nodes (see Table 3.1, MSE explained in Validation section). One possible explanation for this is that the basic recurrent network's representational power is already sufficient to capture the complexities of the gene relationships.

## 3.2 Training Algorithms

Artificial neural networks are commonly trained using backpropagation, a gradient-descent learning algorithm that iteratively refines the weights on each node in the network to approximate the behavior of the training data. However, standard backpropagation cannot be applied to recurrent networks because of the presence of cycles in the network. We could use the Backpropagation Through Time (BPTT) algorithm as described and outlined by Paul Werbos [36], but since BPTT has the tendency to get stuck in local minima, we instead employ the hybrid differential evolution and particle swarm optimization algorithm (DEPSO) used by Xu et al. [39]. Both differential evolution (DE) and particle swarm optimization (PSO) are genetic algorithms that use informed probabilistic behavior to converge on a solution much faster than classic backpropagation.

### Particle Swarm Optimization

Particle swarm optimization (PSO) represents a set of candidate solutions as a swarm of particles in solution space. Each particle $i$ represents a set of values for the network parameters, described by its position $x_i$, and travels through solution space with velocity $v_i$. As the particle travels, it moves toward its best solution and the best solution of the entire swarm population. Equations 3.4 and 3.5 describe this process.

$$v_i(t) = w \times v_i(t-1) + c_i \times \phi_1 \times (p_i - x_i(t-1)) + c_2 \times \phi_2 \times (p_g - x_i(t-1)), \tag{3.4}$$

$$x_i(t) = x_i(t-1) + v_i(t), \tag{3.5}$$

where $w$ is the inertia weight, $c_1$ and $c_2$ are cognitive and social acceleration constants, and $\phi_1$ and $\phi_2$ are randomly drawn from a uniform distribution in the range [0, 1].

The algorithm for PSO in this context proceeds as follows [39]:

(i) Initialize a population of particles with random positions and velocities of $D(= N(N+2))$ dimensions. Specifically, the connection weights, biases, and time constants are randomly generated with uniform probabilities over the range $[w_{min}, w_{max}]$, $[\beta_{min}, \beta_{max}]$, and $[\tau_{min}, \tau_{max}]$, respectively. Similarly, the velocities are randomly generated with uniform probabilities in the range $[-V_{max}, V_{max}]$, where $V_{max}$ is the maximum value of the velocity allowed.

(ii) Calculate the estimated gene expression time series based on the RNN model, and evaluate the optimization fitness function for each particle.

(iii) Compare the fitness value of each particle $Fit(x_i)$ with $Fit(p_i)$. If the current value is better, reset both $Fit(p_i)$ and $p_i$ to the current value and location.

(iv) Compare the fitness value of each particle $Fit(x_i)$ with $Fit(p_g)$. If the current value is better, reset both $Fit(p_g)$ and $p_g$ to the current value and location.

(v) Update the velocity and position of the particles with Equations 3.4 and 3.5.

(vi) Return to step (ii) until a stopping criterion is met.

We set the fitness $Fit(x_i)$ of particle $x_i$ equal to $MSE(x_i)$. Following the conclusions of Xu et al. [39], we set $w$ to $1 - rand()/2$, $c_1$ to 2.5, and $c_2$ to 1.5, which were the optimal parameter values that they found in their experiments.

**Differential Evolution**

Differential evolution (DE) works by evolving individual solutions based on differences between other random solutions selected from the existing population. Evolution causes the population to draw together to convergence. The mutation operator in differential evolution is shown in Equations 3.6 and 3.7.

$$y_{1j} = x_{4j} + \gamma(x_{2j} - x_{3j}), \tag{3.6}$$

$$y_{1j} = x_{1j}, \tag{3.7}$$

where $j$ is the dimension that is to be mutated based on the mutation probability $p_r$ and $\gamma$ is a scaling factor.

The algorithm for differential evolution in this context proceeds as follows [39]:

(i) Initialize a population of M individuals. Set the values of $p_r$ and $gamma$.

(ii) In every generation, for each individual select three distinct individuals randomly from the remaining population.

(iii) For every dimension/parameter of the individual, if mutation is to take place based on the probability $p_r$, Equation 3.6 is used; otherwise Equation 3.7 is used to update the parameter values of the offspring.

(iv) For each parent and its offspring, the individual with the higher fitness is passed on to the next generation.

(v) Repeat steps (iii)-(iv) until all the individuals have satisfied some convergence criterion.

Again informed by the work of Xu et al. [39], we set $p_r$ to 0.3 and $\gamma$ to 0.5.

#### Hybrid Differential Evolution and Particle Swarm Optimization

The hybrid algorithm used by Xu et al. [39] combines PSO and DE as follows:

(i) For every odd iteration, carry out the canonical PSO operation on each individual of the population by implementing steps (i)-(vi) from the PSO algorithm.

(ii) For every even iteration, carry out the DE operation on every particle to create the offspring. Once the offspring is created, their fitness is evaluated against that of the parent. The one with the higher fitness is selected to participate in the next generation.

(iii) The $p_g$ and $p_i$ of the new population are recalculated.

(iv) Repeat steps (i)-(iii) until convergence.

We define the stopping criteria as MSE falling below 0.001 (normalized) or the algorithm exceeding 1000 iterations, whichever occurs first.

## 3.3 Predictions

We investigate two types of prediction methods to evaluate the performance our model. First, we treat each time step as a separate test instance and predict one step into the future using

Equation 3.1. This is the method most commonly used in the literature [19, 20, 26]. Second, in order to evaluate long-term prediction and steady states, we input initial conditions to the model and run it using its own outputs as inputs for subsequent time steps. This method sheds light on the extent of prediction drift as well as the applicability of the model to prediction of novel trajectories.

Throughout the remainder of this study, we will refer to the two prediction methods respectively as ”one-step prediction” and ”long-term prediction”.

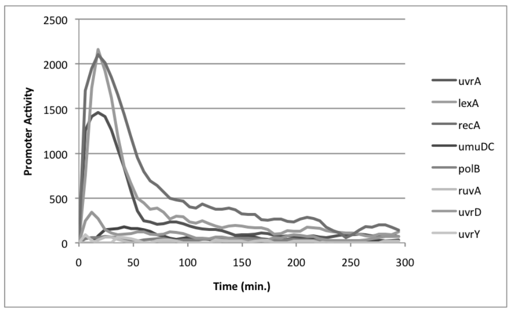
## 3.4 Datasets

We have chosen two datasets from real biological processes in order to evaluate the expressive power and predictive capability of our model. However, since no true biological networks are fully known to researchers and real-world measurements tend to be quite noisy, we have also chosen two synthetic datasets for which we have a “true” solution to compare against.

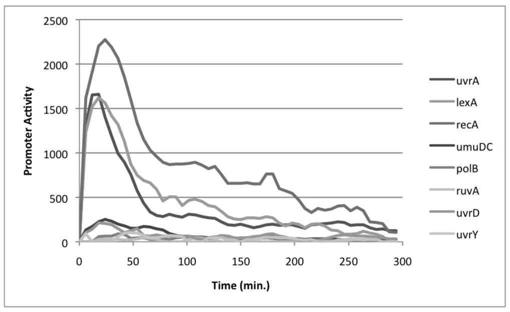
### SOS DNA Repair network in bacterium *Escherichia coli*

The SOS DNA repair system of *Escherichia coli* is a well-characterized transcriptional network frequently used in the literature [19, 20, 26]. It is a single input module consisting of about 30 operons regulated by master repressor *LexA* (see Fig. 3.4. SOS protein *RecA* senses DNA damage and regulates *LexA*, allowing other genes to be expressed. Once damage has been repaired or bypassed, *RecA*'s activation decreases, *LexA* increases, and all cells return to their original state.

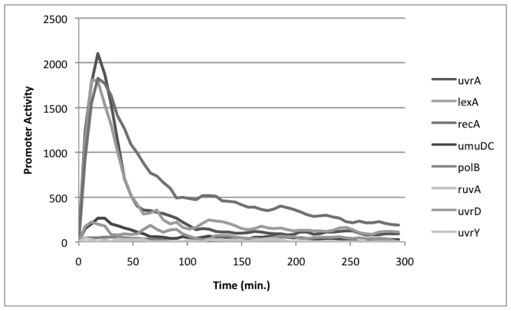
Data for all of the experiments was taken by irradiating the cultures with UV light and taking measurements for two cell cycles. Experiments 1 and 2 were irradiated by UV of 5 $Jm^{-2}$ and experiments 3 and 4 were irradiated by UV of 20 $Jm^{-2}$. The data is normalized by the maximal activity for each operon, and graphs of the time series are shown in Fig. 3.3. We use the data from the peaks onward in order to study recovery from perturbation.



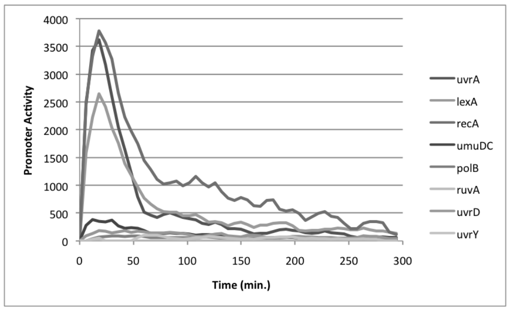
(a) SOS Experiment 1



(b) SOS Experiment 2



(c) SOS Experiment 3



(d) SOS Experiment 4

Figure 3.3: SOS DNA Repair dataset

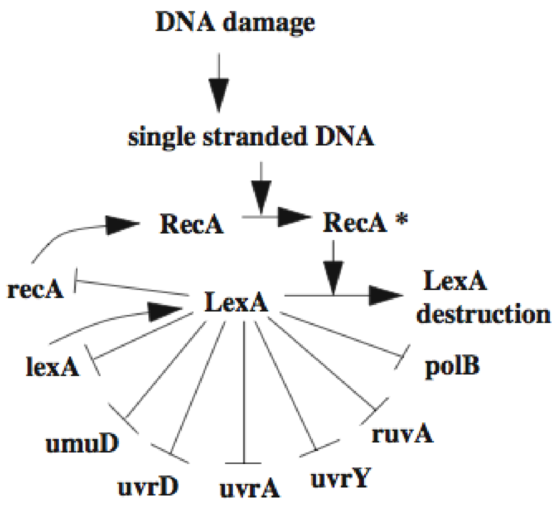


Figure 3.4: SOS DNA Repair network

We have included this dataset to demonstrate our model's capability to describe real biological processes and to compare quantitatively against Ronen and Maraziotis.

### Stanford Yeast Database

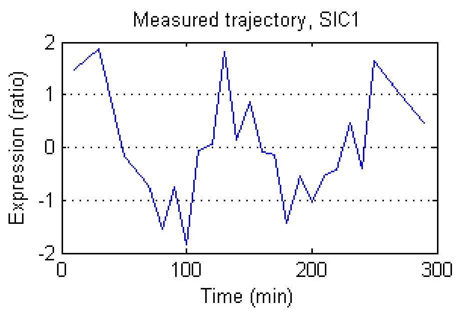
This database contains gene expression measurements taken during the cell cycle of the yeast *Saccharomyces cereviciae* [29]. The experiments contain 59 samples collected at various points in the cell cycle. There are three sets of measurements, named according to the synchronization method that was used for each: *cdc15 arrest* (24 samples), *cdc28 arrest* (17 samples), and *alpha-factor* (18 samples). The *cdc15* and *alpha* datasets were generated using the same experimental setup, while the *cdc28* dataset comes from a previous study [29]. Maraziotis et al. chose a subset of 12 genes identified as highly regulated in previous biological studies, and they filled in missing samples using an estimation method [20]. Sample gene behaviors are shown in Fig. 3.5.

We have included this dataset to further demonstrate our model's capability to represent real biological processes and to compare quantitatively against Maraziotis et al.

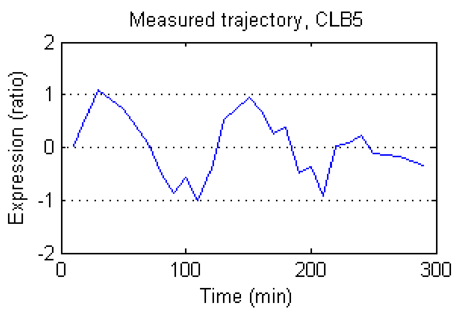
### DREAM3 In-Silico-Network challenge

The fourth challenge of the third Dialogue for Reverse Engineering Assessments and Methods (DREAM3) competition required participants to infer a genetic regulatory network model from time series data and steady state values [10]. The time series data was produced by a synthetic genetic network generated by Daniel Marbach's GeneNetWeaver [22].

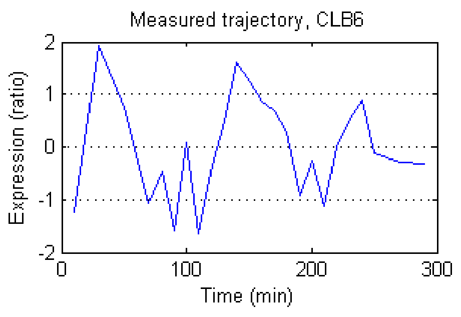
We used the first *E. Coli* network, which consists of 10 genes each. The data consists of wild-type steady states, heterozygous knockdown steady states, null mutant steady states, and four perturbation trajectories for the network. Of these, we were able to use the wild-type steady states and the perturbation trajectories to infer a model for the network. The perturbation trajectories are shown in Fig. 3.6.



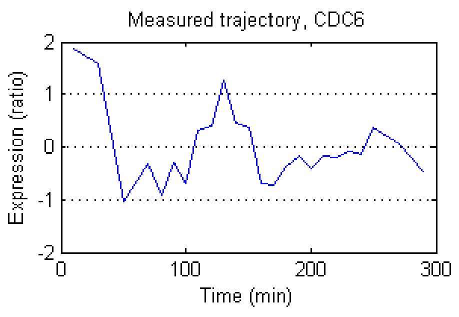
(a) SIC1 trajectory, *cdc15* dataset



(b) CLB5 trajectory, *cdc15* dataset

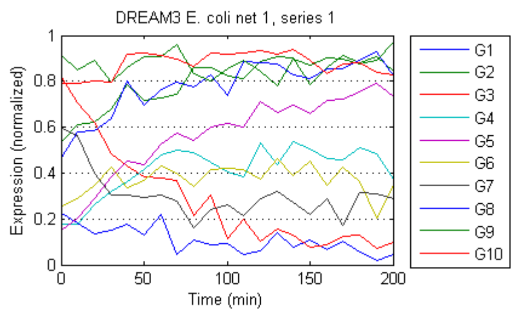


(c) CLB6 trajectory, *cdc15* dataset

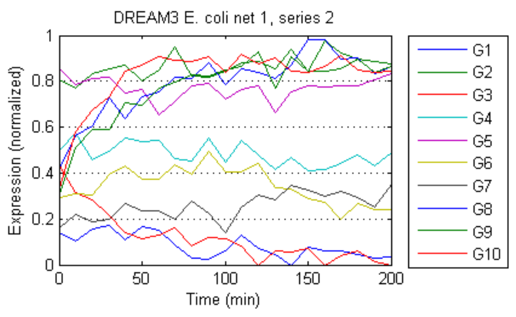


(d) CDC6 trajectory, *cdc15* dataset

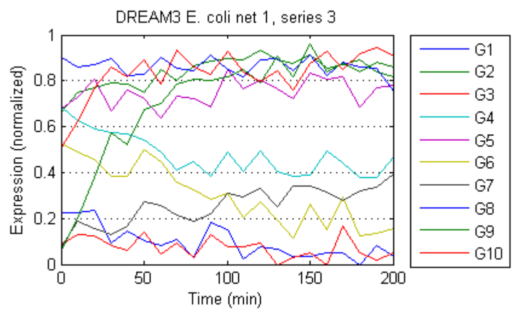
Figure 3.5: Stanford Yeast Database dataset



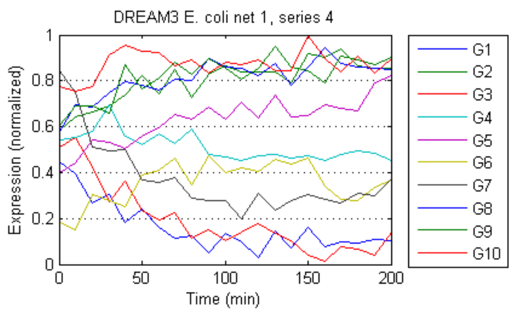
(a) DREAM3 *E. coli* dataset, series 1



(b) DREAM3 *E. coli* dataset, series 2



(c) DREAM3 *E. coli* dataset, series 3



(d) DREAM3 *E. coli* dataset, series 4

Figure 3.6: DREAM3 dataset

We have included this dataset to study the performance of our system on networks whose true structures are known but whose dynamics are calculated using a different model than the one under study. Also, unlike the other datasets, this dataset contains a variety of initial conditions that produce different qualitative profiles, allowing us to evaluate the predictive ability of our system.

### DREAM4 In Silico challenge

The second challenge of the fourth Dialogue for Reverse Engineering Assessments and Methods (DREAM4) competition was quite similar to the fourth challenge of the previous year [10, 11]. There are five synthetic networks, again generated by Daniel Marbach's GeneNetWeaver [22] and consisting of 10 genes each. The same types of data are presented with the addition of multifactorial perturbation steady states, and a bonus round challenges competitors to predict the steady state values for dual knockout strains (where two genes are never expressed).

We have included this dataset to again study the performance of our system on networks whose true structures are known but whose dynamics are calculated using a different model than the one under study. We also use it to test prediction of steady states and to compare quantitatively against DREAM4 competitors.

# Chapter 4

# Validation and Results

The greatest challenge to genetic network modeling is that biologists do not know the entire truth behind any real genetic network. Because no complete topologies are known, inferred topologies for real biological networks cannot be directly evaluated as correct or incorrect. Synthetic networks can be used to simulate and evaluate the ability of inference systems to recover unknown networks, but successful recovery of a synthetic network does not necessarily translate to success on real biological networks.

Relevancy to drug discovery absolutely demands ecological validity; this motivates our use of gene expression prediction as an indicator of the validity of a solution. Although it is impossible to completely and confidently evaluate the topologies generated by a model, it is quite possible to evaluate future behavior predicted by that model against real data.

## 4.1 Metrics

We present here our evaluation metrics for model training, comparison to previous methods, and viable ways to evaluate predictive behavior in a genetic network model.

### Mean squared error

Our objective for training the network is minimization of mean squared error (MSE), which measures how much the network output $e(t)$ differs from the target output $d(t)$. MSE is defined by Equation 4.1.

$$MSE = \frac{1}{TN}\sum_{t=0}^{T}\sum_{i=1}^{N}(e_i(t) - d_i(t))^2 \tag{4.1}$$

MSE can be interpreted as a measure of how closely our system models the particular expression values in a particular dataset. As a precaution against overfitting, we limit the number of training iterations on each dataset.

### Mean error

Mean error is similar to MSE, but measures normalized error, shown in Equation 4.2. It is utilized by both Ronen and Maraziotis to evaluate the success of their models [26, 20, 21]. However, two significant weaknesses of this metric evidenced by Equation 4.2 are its high sensitivity to very small values and its inability to handle zero values. Microarray data often contains values that are very close to or equal to zero, and the level of noise that is usually present in microarray measurements may overwhelm the usefulness of this metric.

$$ME_i = \frac{1}{T}\sum_{t=0}^{T}\frac{|e_i(t) - d_i(t)|}{d_i(t)} \tag{4.2}$$

However, despite the potential flaws in using such a metric, we have elected to measure mean error on our system for the purpose of comparison with previous methods.

### Prediction residuals

In order to evaluate prediction error, we measure the absolute value of the prediction residual of each gene $i$ as a function of time $t$ using Equation 4.3.

$$Resid_i(t) = |e_i(t) - d_i(t)| \tag{4.3}$$

Plotting prediction residual versus time provides a visual description of prediction error for both one-step and long-term prediction, and prediction drift for long-term prediction.

## Magnitude squared coherence

While prediction residuals are useful for graphical representation of prediction drift, they do not satisfy the need for a comprehensive measure of prediction success. We have thus decided to use magnitude squared coherence as an additional error metric to describe correlation between predicted time series and target trajectories.

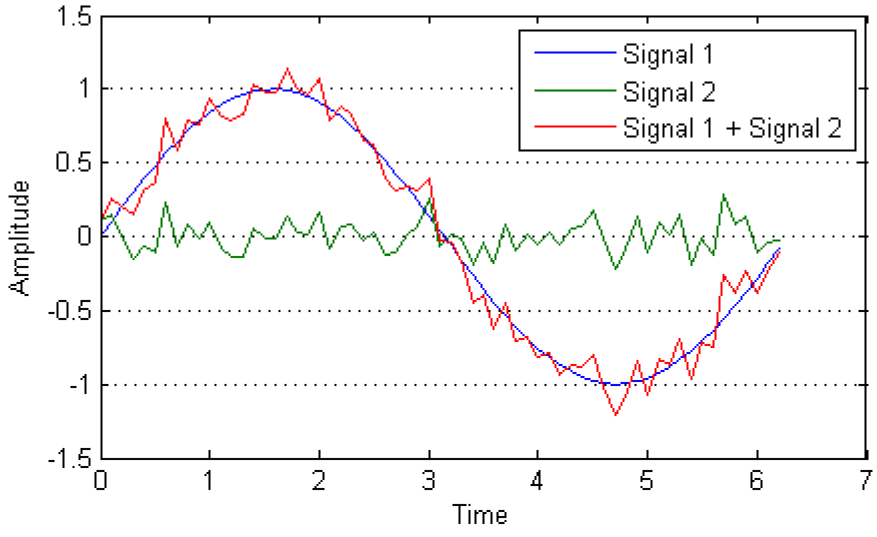
Magnitude squared coherence can be used as a measure of correlation between two signals and is defined by Equation 4.4, where $P_{xx}$ is the power spectral density of signal $x$ and $P_{xy}$ is the cross-spectral density of signals $x$ and $y$. From this point forward, we will refer to magnitude squared coherence simply as coherence.

$$C_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)} \tag{4.4}$$

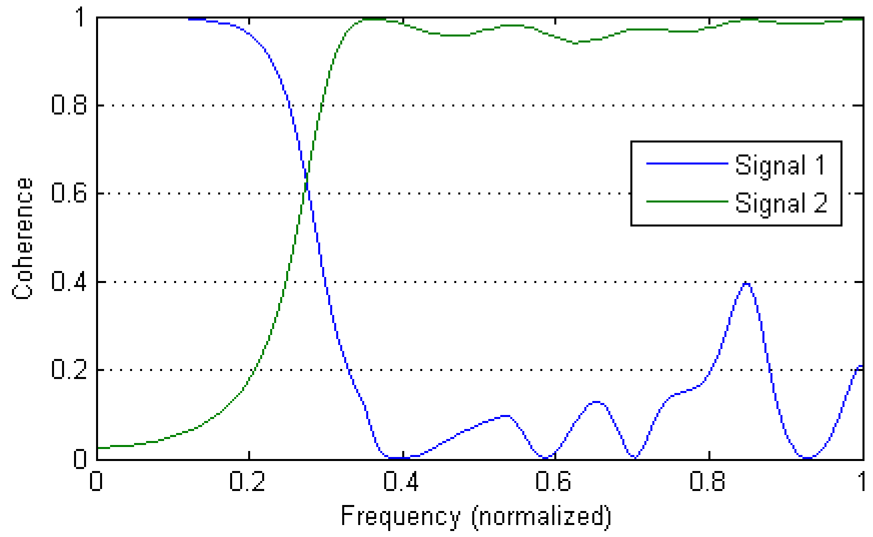
Coherence is a phase-independent measure of the cross-correlation between signals with respect to spectral density. It ranges from 0 to 1, where higher values correspond to higher correlation between two signals. Coherence values at lower frequencies can be interpreted as an indicator of how successfully a model captures long-term trends. On the other hand, coherence in high frequencies indicates how successfully the model captures short-term details. Because gene expression data is quite noisy due to errors in the collection process, coherence at high frequencies may not be reliable as a measure of the accuracy of a long-term prediction.

A simple example of magnitude squared coherence is shown in Fig. 4.1. In Fig. 4.1(a), we have composed a noisy signal by adding a smooth sine curve (Signal 1) and a signal generated from a normal distribution with 0.3 standard deviation (Signal 2). Because of the nature of the composition, Signal 1 is representative of the smooth, long-term trend of the composite signal, and Signal 2 is representative of the noisy details of the composite signal.

Fig. 4.1(b) shows the coherence values between each original signal and the composite signal. Here, it is evidenced that coherence in the lower frequencies correspond to long-term



(a) Two signals and their sum



(b) Coherence between each signal and the sum of the two

Figure 4.1: Example of magnitude squared coherence

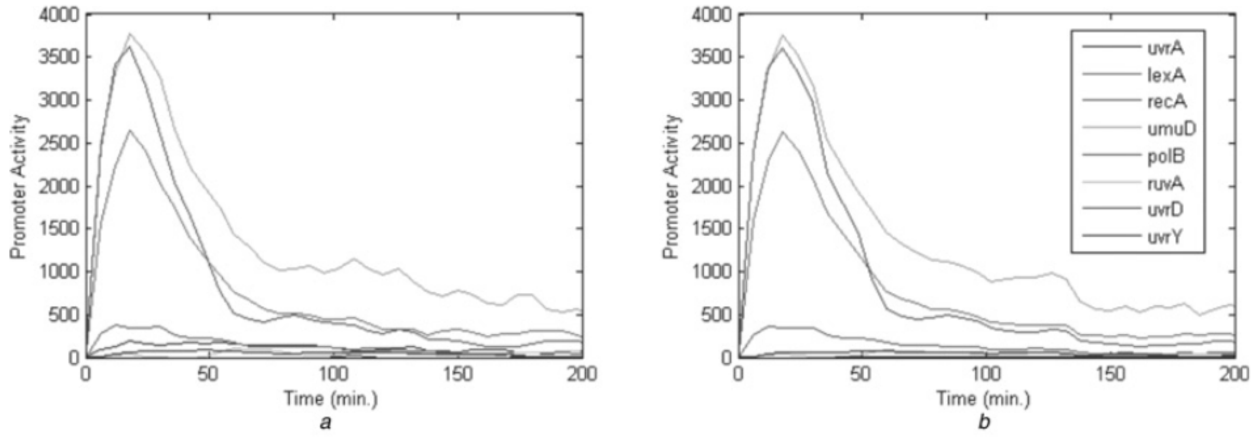


Figure 4.2: Actual data from SOS experiment 2 (a) vs prediction of SOS experiment 2 using Maraziotis recurrent neurofuzzy network (b) [21]

trends while coherence in the higher frequencies correspond to shorter-term details in the signal. For the application at hand, although high coherence across all frequencies is ideal, we consider the lower-frequency content to be relatively more significant in evaluating prediction accuracy than the higher-frequency content.

Butte et al. reported success when using coherence as a measure of similarity between expression profiles in order to identify co-regulated genes [5]. Their method was able to recover valid biological relationships between genes that are not found by using MSE or Pearson's correlation coefficient due to shifts in phase between the trajectories. These results motivate our investigation of coherence as a useful evaluation of predicted trajectories.

## 4.2 Results

### Biological modeling capability

Figs. 4.2 and 4.3 show the results of one-step prediction on SOS experiment 2 by both Maraziotis' recurrent neurofuzzy network [21] and our implementation of DEPSO-RNN. Table 4.1 shows the mean error values for each gene and overall mean error for each model.

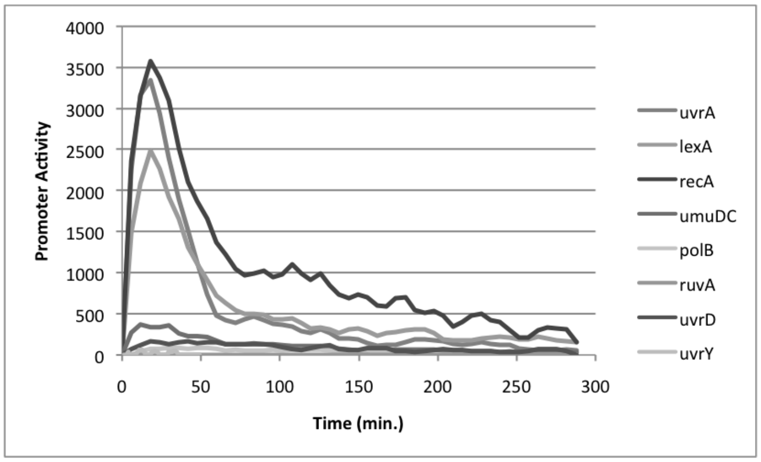


Figure 4.3: Prediction of SOS experiment 2 using DEPSO-RNN

Table 4.1: Comparison of mean error for SOS Experiment 1

| Gene | Ronen | Maraziotis | Chan |
|---|---|---|---|
| uvrA | 0.14 | 0.090 | 0.128 |
| lexA | 0.10 | 0.084 | 0.077 |
| recA | 0.12 | 0.100 | 0.043 |
| umuD | 0.21 | 0.085 | 0.147 |
| polB | 0.31 | 0.079 | 0.261 |
| ruvA | 0.22 | 0.204 | 0.494 |
| uvrD | 0.20 | 0.172 | 0.190 |
| uvrY | 0.45 | 0.16 | 0.388 |
| Average | *0.22* | *0.122* | *0.216* |

Table 4.2: Comparison of mean error for SOS Experiment 2

| Gene | Maraziotis | Chan |
|---|---|---|
| uvrA | 0.115 | 0.069 |
| lexA | 0.105 | 0.092 |
| recA | 0.120 | 0.041 |
| umuD | 0.200 | 0.094 |
| polB | 0.302 | 0.342 |
| ruvA | 0.201 | 0.267 |
| uvrD | 0.195 | 0.200 |
| uvrY | 0.420 | 0.364 |
| Average | *0.207* | *0.184* |

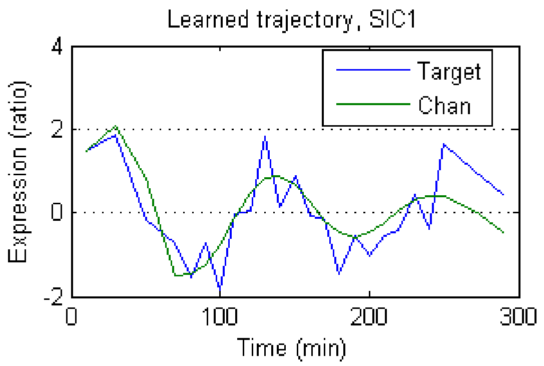
For SOS experiment 1, we achieve lower overall mean error than Ronen et al., whose method is based on biologically informed kinetic equations. This suggests that our model is well able to describe real biological behavior. Although our model does not achieve as low mean error as Maraziotis et al. for experiment 1, Table 4.2 shows that our model is better able to generalize to experiment 2. This may indicate overfitting in Maraziotis' model.

Maraziotis et al. did not report their mean error results for SOS experiments 3 and 4, noting that their method was unable to achieve adequate results. This is likely due to the difference in perturbation magnitude between SOS experiments 3 and 4 and the other two experiments; experiments 3 and 4 contain values that fall well outside the range of the training data. However, our method was able to achieve comparable results to experiment 2, which are reported in Table 4.3. The exception to this success was prediction of *ruvA*, which resulted in very small values that caused mean error to explode in magnitude.

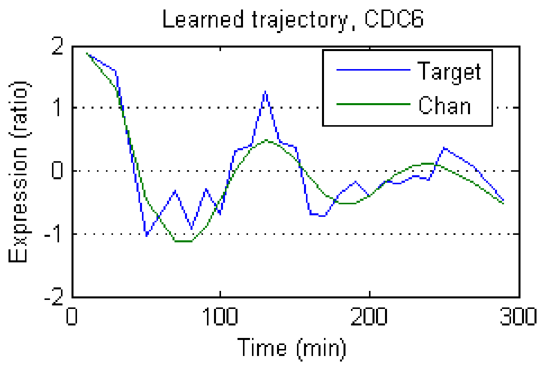
Table 4.3: Mean error results for all SOS test datasets using DEPSO-RNN

| Gene | Exp 2 | Exp 3 | Exp 4 |
|---|---|---|---|
| uvrA | 0.069 | 0.103 | 0.123 |
| lexA | 0.092 | 0.114 | 0.054 |
| recA | 0.041 | 0.026 | 0.044 |
| umuD | 0.094 | 0.172 | 0.171 |
| polB | 0.342 | 0.145 | 0.117 |
| ruvA | 0.267 | - | - |
| uvrD | 0.200 | 0.238 | 0.126 |
| uvrY | 0.364 | 0.320 | 0.534 |
| Average | *0.184* | *0.160* | *0.167* |

Fig. 4.4 shows two examples of trajectories that were learned by our model for the Stanford Yeast Database training dataset. Although the original data is quite noisy, our model is able to describe the longer-term trends that it follows. Table 4.4 shows the mean error values and overall mean error for both our model and Maraziotis' recurrent neurofuzzy network on the test datasets. Again, we were able to achieve better overall generalization than Maraziotis. This gives confidence in the biological modeling capability of our model.



(a) SIC1 trajectory, *cdc15* dataset



(b) CDC6 trajectory, *cdc15* dataset

Figure 4.4: Learned trajectories for Stanford Yeast Database training set

Table 4.4: Comparison of MSE for Stanford Yeast Database test sets

| Gene | Maraziotis(alpha) | Chan(alpha) | Maraziotis(cdc28) | Chan(cdc28) |
|---|---|---|---|---|
| SIC1 | 0.744 | 0.117 | 0.406 | 0.189 |
| CLB5 | 0.446 | 0.166 | 0.177 | 0.240 |
| CDC20 | 0.619 | 0.115 | 0.366 | 0.161 |
| CLN3 | 0.149 | 0.192 | 0.247 | 0.239 |
| SWI6 | 0.498 | 0.072 | 0.331 | 0.061 |
| CLN1 | 0.665 | 0.465 | 0.364 | 0.259 |
| CLN2 | 0.735 | 0.709 | 0.575 | 0.449 |
| CLB6 | 0.252 | 0.528 | 0.365 | 0.620 |
| CDC28 | 0.058 | 0.216 | 0.068 | 0.110 |
| MBP1 | 0.699 | 0.090 | 0.429 | 0.081 |
| CDC6 | 0.424 | 0.155 | 0.337 | 0.275 |
| SWI4 | 0.122 | 0.359 | 0.490 | 0.292 |
| Average | *0.451* | *0.265* | *0.346* | *0.248* |

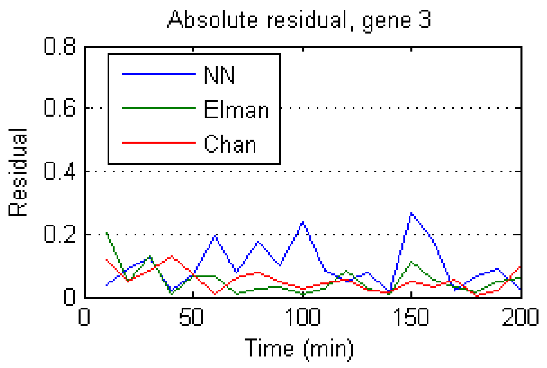
**Prediction capability**

In order to evaluate prediction capability, we used datasets from DREAM3 and DREAM4 to ensure that true trajectories were known. For the DREAM3 dataset, we trained our model on the first three trajectories and tested it on the fourth. All four trajectories consisted of different initial conditions allowing for truly unknown testing data. Based on the work of Smith et al., who previously concluded that feedforward neural networks with hidden layers and simple Elman recurrent neural networks achieve the best short-term prediction accuracy of all simple neural networks [28], we also generated predictions using both a feedforward neural network with hidden layers and an Elman network. Fig. 4.5 shows the results of both one-step and long-term prediction for three genes in the fourth trajectory.

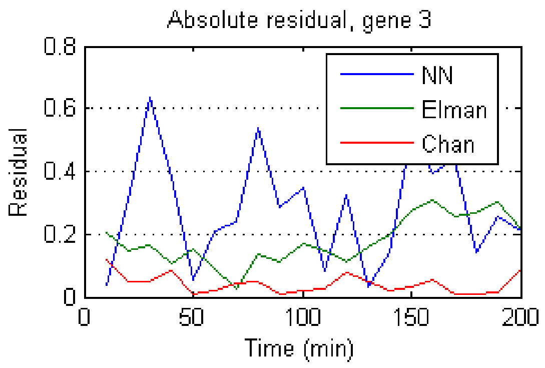
In Figs. 4.5(a), 4.5(c) and 4.5(e), the one-step prediction results show comparable performance among the three models. The residual for some of the prediction points for the neural network is significantly higher than at others; this is most likely caused by overfitting to the training data. Because the original data is noisy, the neural network has learned to predict specific sets of values that may change in completely opposite directions due to the noise oscillation. The Elman network performs somewhat better due to the element of recurrence in its structure, which helps to control for sudden changes in value.

One-step prediction does not allow for observation of prediction drift because each time step is treated as a separate test instance. This is limiting because models that are sensitive to new initial conditions will appear to do better at subsequent time points. For example, the neural network does not perform well in the first few data points in Figs. 4.5(c) and 4.5(e), but gets "back on track" on later data points. However, in the long-term, initial errors could potentially cause serious prediction drift later on.

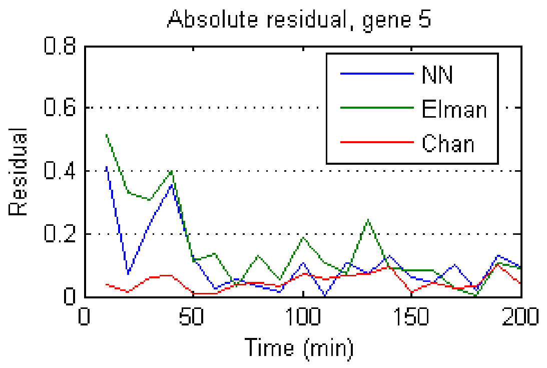
Figs. 4.5(b), 4.5(d) and 4.5(f) show the effect of prediction drift in long-term prediction. Both the neural network and the Elman network perform significantly worse in long-term prediction than in one-step prediction due to sensitivity to unseen values and accumulated errors. However, our model continues to perform quite well in the long-term.



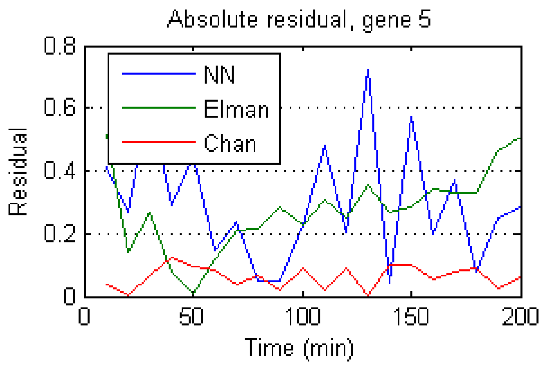
(a) One-step prediction, gene 3



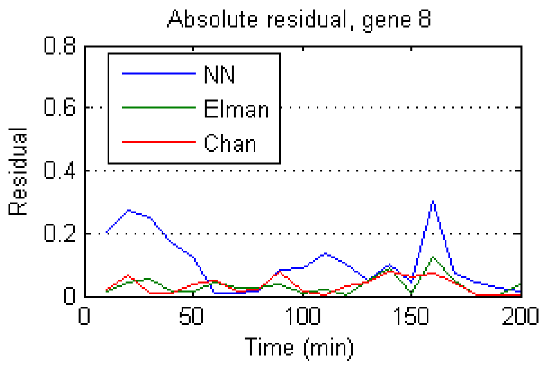
(b) Long-term prediction, gene 3



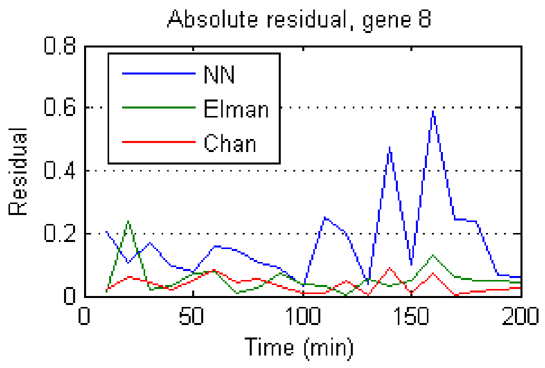
(c) One-step prediction, gene 5



(d) Long-term prediction, gene 5

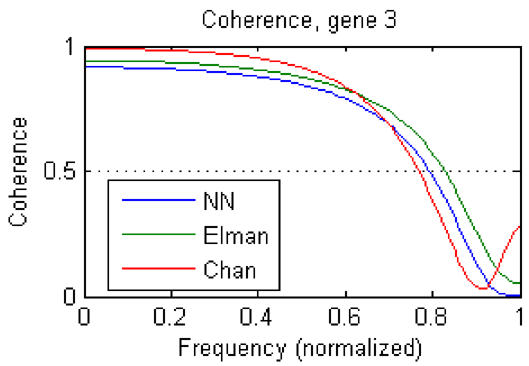


(e) One-step prediction, gene 8

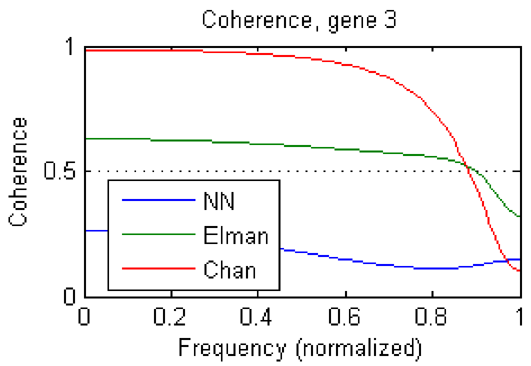


(f) Long-term prediction, gene 8

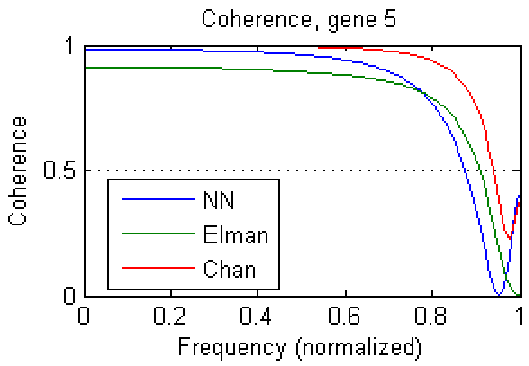
Figure 4.5: Comparison of one-step (a,c,e) and long-term (b,d,f) prediction



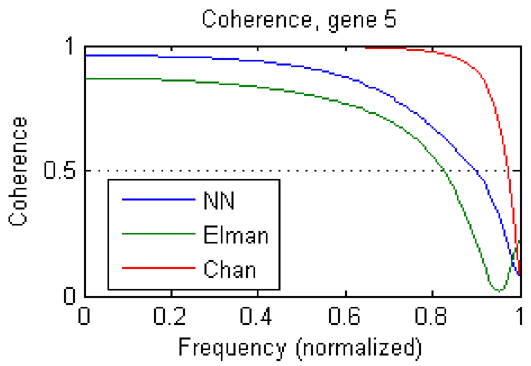
(a) One-step prediction, gene 3



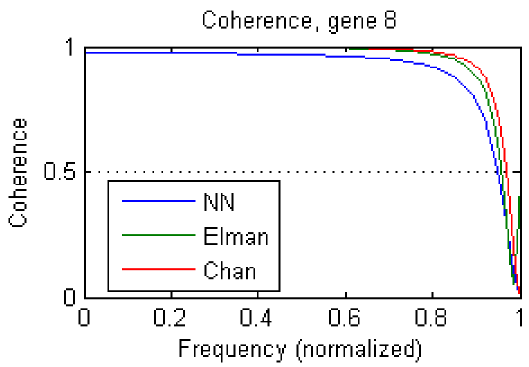
(b) Long-term prediction, gene 3



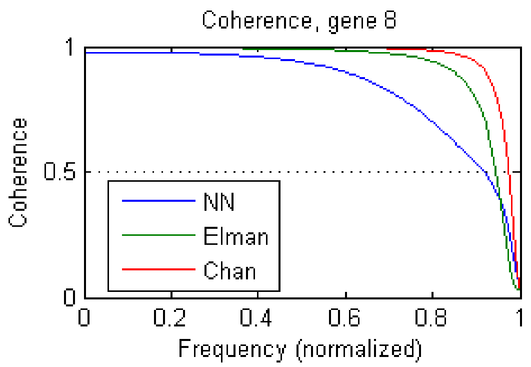
(c) One-step prediction, gene 5



(d) Long-term prediction, gene 5



(e) One-step prediction, gene 8



(f) Long-term prediction, gene 8

Figure 4.6: Comparison of coherence for one-step (a,c,e) and long-term (b,d,f) prediction

In order to capture these observations in a comprehensive metric, we evaluated coherence for both one-step and long-term predictions on all three models. The results are shown in Fig. 4.6. Recall from section 4.1 that higher values of coherence at low frequencies indicate higher correlation in overall signal shape, and higher values of coherence at high frequencies indicate higher correlation in smaller-scale signal fluctuations. Figs. 4.6(a), 4.6(c) and 4.6(e) show that coherence is similar for all three models when one-step prediction is used. This makes sense because even a naive system that predicts no change from one step to the next would exhibit a coherence of 1 at all frequencies due to the perfect replication of the trajectory one step shifted. Hence, coherence is not useful for evaluation of one-step predictions such as those computed by Maraziotis et al.

However, Figs. 4.6(b), 4.6(d) and 4.6(f) show that coherence is a useful description of the overall prediction performance of a model in the long term. It is able to differentiate between our model, which adheres to the overall shape of the trajectories, and the other two models, which suffer from prediction drift in the long term.

For the DREAM4 dataset, we generated a time series for the wild-type steady state and used it to train our model along with the perturbation trajectories. We then simulated knockouts by fixing expression values to zero for the genes in question. Table 4.5 show the results of our steady state predictions compared to other DREAM4 competitors.

Table 4.5: Comparison of MSE in DREAM4 competition

| Team | MSE overall | MSE 1 | MSE 2 | MSE 3 | MSE 4 | MSE 5 |
|---|---|---|---|---|---|---|
| Team 543 | *0.015* | 0.008 | 0.029 | 0.025 | 0.009 | 0.003 |
| Team 532 | *0.023* | 0.021 | 0.051 | 0.013 | 0.009 | 0.024 |
| Team 548 | *0.044* | 0.039 | 0.035 | 0.029 | 0.012 | 0.104 |
| Team 498 | *0.044* | 0.029 | 0.043 | 0.078 | 0.015 | 0.055 |
| Chan | *0.046* | 0.036 | 0.044 | 0.058 | 0.020 | 0.076 |
| Team 522 | *0.053* | 0.030 | 0.023 | 0.071 | 0.014 | 0.126 |
| Team 347 | *0.118* | 0.152 | 0.097 | 0.118 | 0.072 | 0.149 |
| Team 236 | *0.155* | 0.161 | 0.179 | 0.138 | 0.128 | 0.168 |

Although our model does not show the best performance of the group, it is competitive with the majority of the teams. This is despite the fact that our model could not utilize the additional single knockout data that was provided due to the need for time series data. Our model is additionally able to produce a full time trajectory for arrival at these steady states, showing not only the change in steady state, but the time scale over which the change occurs. This is a feature that no other models that we know of have the ability to produce.

# Chapter 5

# Conclusion

## 5.1 Discussion

We have shown that recurrent neural networks are capable of modeling genetic network dynamics in real biological systems by using it to accurately model both the SOS DNA repair network in the bacterium *Escherichia coli* and the cell cycle of the yeast *Saccharomyces cereviciae*. In addition, we have shown that it achieves better generalization on unseen conditions through its ability to attain lower overall mean error on the unseen test sets than state-of-the-art methods such as kinetic parameter estimation and recurrent neurofuzzy networks [26, 21].

Our model's superior long-term prediction accuracy is demonstrated by its ability to achieve lower prediction residuals and higher coherence than other models on the DREAM3 dataset, which contains significantly differing initial conditions among the trajectories. Our model outperforms both feedforward neural networks with hidden layers and Elman recurrent networks, which were shown by Smith et al. to be the best-performing predictive neural networks in the short term [28]. While the latter two types of models suffer from overfitting and prediction drift, our model is able to avoid both weaknesses and accurately predict future behavior. Also, our model is able to compete with other methods in predicting steady state levels on the DREAM4 dataset.

With all of this evidence brought to bear, we conclude that recurrent neural networks are the best method today for modeling genetic networks with a view towards application to drug discovery.

## 5.2 Contributions

We have demonstrated that our modification of the DEPSO-RNN paradigm of Xu et al. [39] is capable of describing real biological systems, generalizing to unseen conditions, and predicting future behavior. Our system has the advantage of rapid training time, long-term vision, ability to handle time steps of varying lengths, and avoidance of overfitting to specific data points. In addition, the absence of hidden layers preserves the interpretability of the model; weights between nodes can be directly read as causal relationships.

We have also shown that there is a great need for improved validation methods in the area of genetic network modeling in order to successfully achieve cost and time reduction in drug discovery and development. To encourage this, we have provided a new validation framework: we have demonstrated the usefulness of gene expression prediction in the evaluation of generalization in genetic inference systems; we have drawn a distinction between the one-step prediction method that is often used throughout the literature and the long-term prediction method that we advocate for accurate, meaningful evaluation of a predictive model; and we have called attention to a comprehensive metric, magnitude squared coherence, to be used as a measurement of the success of long-term predictions.

The relationships among genes are an essential key to unlocking the biological mysteries of life. The ability to decipher genetic regulatory networks and predict gene expression levels in response to outside stimuli will universally impact the quality of life both through superior health care and through the advancement of life science research. Successful gene network inference and gene expression prediction will greatly expedite medical diagnosis as well as drug design and development for therapeutic treatment. Adverse conditions will be better identified and treated prior to manifestation, and natural, personalized medicines may someday be engineered to stimulate unbalanced regulatory networks towards self-correction. Success of this research will improve the quality of life for society worldwide.

In addition to applied medical benefits, successful gene network inference and gene expression prediction will vastly further the fields of both biology and computer science.

In biology, the understanding of how gene networks operate will be a treasure trove of knowledge. Also, with knowledge of underlying network structure, many more experiments can be conducted *in silico* and the results quickly processed rather than waiting for weeks for results from a wet lab. In computer science, many important precedents will be formed for the difficult problem of choosing and ranking among infinite well-fitting hypotheses and incorporating prior knowledge in order to help refine the search through hypothesis space.

## References

[1] Z. Bar-Joseph. *Analyzing Time Series Gene Expression Data*, Bioinformatics, vol. 20, no. 16, pp. 2493-2503, 2004.

[2] N. Barker, C. Myers and H. Kuwahara. *Improved Algorithm for Learning of Genetic Regulatory Network Connectivity from Time Series Data*, Transactions on Computational Biology and Bioinformatics, vol. 8, 2007.

[3] S. Bhadra, C. Bhattacharyya, N. R. Chandra and I. S. Mian. *A Linear Programming Approach for Estimating the Structure of a Sparse Linear Genetic Network from Transcript Profiling Data*, Algorithms for Molecular Biology, vol. 4, no. 5, 2009.

[4] J. R. Bock and D. A. Gough. *Whole-Proteome Interaction Mining*, Bioinformatics, vol. 19, no. 1, pp. 125-135, 2003.

[5] A. J. Butte, L. Bao, B. Y. Reis, T. W. Watkins and I. S. Kohane. *Comparing the Similarity of Time-Series Gene Expression Using Signal Processing Metrics*, Journal of Biomedical Informatics, vol. 34, pp. 396-405, 2001.

[6] T. Chen, H. L. He and G. M. Church. *Modeling Gene Expression with Differential Equations*, Pacific Symposium on Biocomputing, pp. 29-40, 1999.

[7] X. Dai. *Genetic Regulatory Systems Modeled by Recurrent Neural Network*, Lecture Notes in Computer Science: Advances in Neural Networks, vol. 3174, pp. 519-524, 2004.

[8] M. S. Dasika, A. Gupta and C. D. Maranas. *A Mixed Integer Linear Programming (MILP) Framework for Inferring Time Delay in Gene Regulatory Networks*, Pacific Symposium on Biocomputing, pp. 474-486, 2004.

[9] H. de Jong. *Modeling and Simulation of Genetic Regulatory Systems: A Literature Review*, Journal of Computational Biology, vol. 9, no. 1, pp. 67-103, 2002.

[10] DREAM Initiative. *DREAM3 2008 Reverse Engineering Challenges*, available at http://wiki.c2b2.columbia.edu/dream/index.php/DREAM3conf, accessed Sept 15 2008.

[11] DREAM Initiative. *DREAM4 2009 Reverse Engineering Challenges*, available at http://wiki.c2b2.columbia.edu/dream/index.php/DREAM4conf, accessed Sept 5 2010.

[12] C. Fogelberg and V. Palade. *Machine Learning and Genetic Regulatory Networks: A Review and a Roadmap*, Research Report CS-RR-08-04, Computing Science Group, Oxford University, 2008.

[13] N. Friedman, M. Linial, I. Nachman and D. Peer. *Using Bayesian Networks to Analyze Expression Data*, Journal of Computational Biology, vol. 7, no. 3-4, pp. 601-620, 2000.

[14] C. L. Giles, S. Lawrence and A. C. Tsoi. *Noisy Time Series Prediction Using a Recurrent Neural Network and Grammatical Inference*, Machine Learning, vol. 44, no. 1/2, pp. 161-183, 2001.

[15] P. M. Haverty, U. Hansen and Z. Weng. *Computational Inference of Transcriptional Regulatory Networks from Expression Profiling and Transcription Factor Binding Site Identification*, Nucleic Acids Research, vol. 32, no. 1, pp. 179-188, 2004.

[16] X. Hu, A. Maglia and D. C. Wunsch II. *A General Recurrent Neural Network Approach to Model Genetic Regulatory Networks*, Proceedings of the IEEE Engineering in Medicine and Biology 27th Annual Conference, 2005.

[17] P. Koduru, S. Das, S. M. Welch and J. L. Roe. *Fuzzy Dominance Based Multi-Objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models*, Lecture Notes in Computer Science: Proceedings of the Genetic and Evolutionary Computing Conference, vol. 3102, pp. 356-367, 2004.

[18] W.-P. Lee and K.-C. Yang. *A Clustering-Based Approach for Inferring Recurrent Neural Networks as Gene Regulatory Networks*, Neurocomputing, vol. 71, pp. 600-610, 2008.

[19] R. Manshaei, P. Sobhe Bidari, J. Alirezaie, and M. A. Malboobi. *Time Series Prediction of Gene Expression in the SOS DNA Repair Network of Escherichia coli Bacterium Using Neuro-Fuzzy Networks*, Proceedings of the 13th International Conference on Biomedical Engineering, vol. 23, pp. 1846-1849, 2009.

[20] I. Maraziotis, A. Dragomir and A. Bezerianos. *Recurrent Neuro-fuzzy Network Models for Reverse Engineering Gene Regulatory Interactions*, Lecture Notes in Computer Science: Computational Life Sciences, First International Symposium, pp. 24-34, 2005.

[21] I. Maraziotis, A. Dragomir and A. Bezerianos. *Gene Networks Reconstruction and Time-Series Prediction from Microarray Data Using Recurrent Neural Fuzzy Networks*, IET Systems Biology, vol. 1, no. 1, pp. 41-50, 2007.

[22] D. Marbach. *GeneNetWeaver: DREAM network inference challenge*, available at http://gnw.sourceforge.net/index.html, accessed Sept 15 2008.

[23] P. Mendes, W. Sha and K. Ye. *Artificial Gene Networks for Objective Comparison of Analysis Algorithms*, Bioinformatics, vol. 19 supplement 2, pp. ii122-ii129, 2003.

[24] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet and F. d'Alchè-Buc. *Gene Networks Inference using Dynamic Bayesian Networks*, Bioinformatics, vol. 19 supplement 2, pp. ii138-ii148, 2003.

[25] Pharmaceutical Product Development, Inc. *About Drug Discovery and Development*, available at http://www.ppdi.com/about_ppd/drug_development.htm, accessed Sept 10 2010.

[26] M. Ronen, R. Rosenberg, B. I. Shraiman, and U. Alon. *Assigning Numbers to the Arrows: Parameterizing a Gene Regulation Network by Using Accurate Expression Kinetics*, Proceedings of the National Academy of Sciences, vol. 99, no. 16, pp. 10555-10560, 2002.

[27] I. Shmulevich, E. R. Dougherty, S. Kim and W. Zhang. *Probabilistic Boolean Networks: A Rule-Based Uncertainty Model for Gene Regulatory Networks*, Bioinformatics, vol. 18, no. 2, pp. 261-274, 2002.

[28] M. R. Smith, M. Clement, T. Martinez and Q. Snell. *Time Series Gene Expression Prediction using Neural Networks with Hidden Layers*, Proceedings of the 7th Annual Biotechnology and Bioinformatics Symposium, pp. 67-69, 2010.

[29] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein and B. Futcher. *Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization*, Molecular Biology of the Cell, vol. 9, pp. 3273-3297, 1998.

[30] K. W. Tang and H.-J. Chen. *A Comparative Study of Basic Backpropagation and Backpropagation Through Time Algorithms*, Technical Report TR-700, College of Engineering and Applied Sciences, State University of NY at Stony Brook, 1994.

[31] T. Tian and K. Burrage. *Stochastic Neural Network Models for Gene Regulatory Networks*, Proceedings of the IEEE Congress on Evolutionary Computation, 2003.

[32] J. Vohradsky. *Neural Network Model of Gene Expression*, Federation of American Societies for Experimental Biology, vol. 15, pp. 846-854, 2001.

[33] J. Wang, L. W.-K. Cheung and J. Delabie. *New Probabilistic Graphical Models for Genetic Regulatory Networks Studies*, Journal of Biomedical Informatics, vol. 38, pp. 443-455, 2005.

[34] Y. Wang, T. Joshi, X.-S. Zhang, D. Xu and L. Chen. *Inferring Gene Regulatory Networks from Multiple Microarray Datasets*, Bioinformatics, vol. 22, no. 19, pp. 2413-2420, 2006.

[35] Y. Wang, T. Joshi, D. Xu, X.-S. Zhang and L. Chen. *Supervised Inference of Gene Regulatory Networks by Linear Programming*, Proceedings of the International Conference on Intelligent Computing, vol. 3, pp. 551-561, 2006.

[36] P. J. Werbos. *Backpropagation Through Time: What It Does and How to Do It*, Proceedings of the IEEE, vol. 78, no. 10, pp. 1550-1560, 1990.

[37] R. Xu, X. Hu and D. C. Wunsch II. *Inference of Genetic Regulatory Networks from Time Series Gene Expression Data*, Proceedings of the IEEE International Joint Conference on Neural Networks, 2004.

[38] R. Xu, D. C. Wunsch II and R. L. Frank. *Inference of Genetic Regulatory Networks with Recurrent Neural Network Models Using Particle Swarm Optimization*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 4, no. 4, 2007.

[39] R. Xu, G. K. Venayagamoorthy and D. C. Wunsch II. *Modeling of Gene Regulatory Networks with Hybrid Differential Evolution and Particle Swarm Optimization*, Neural Networks, vol. 20, no. 8, pp. 917-927, 2007.

[40] E. F.-Y. Young and L.-W. Chan. *Using Recurrent Network in Time Series Prediction*, Proceedings of the World Congress on Neural Networks, vol. 4, pp. 332-336, 1993.

[41] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink and E. D. Jarvis. *Advances to Bayesian Network Inference for Generating Causal Networks from Observational Biological Data*, Bioinformatics, vol. 20, no. 18, pp. 3594-3603, 2004.

[42] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. Bittner and E. R. Dougherty. *A Bayesian Connectivity-Based Approach to Constructing Probabilistic Gene Regulatory Networks*, Bioinformatics, vol. 20, no. 17, pp. 2918-2927, 2004.