This document is downloaded from DR-NTU (https://dr.ntu.edu.sg)
Nanyang Technological University, Singapore.

Deep transfer learning for classification of time-delayed Gaussian networks

Chaturvedi, Iti; Ong, Yew Soon; Arumugam, R. V.

2014

Chaturvedi, I., Ong, Y. S., & Arumugam, R. V. (2015). Deep transfer learning for classification of time-delayed Gaussian networks. Signal Processing, 110, 250-262.

https://hdl.handle.net/10356/82815

https://doi.org/10.1016/j.sigpro.2014.09.009

© 2014 Elsevier. This is the author created version of a work that has been peer reviewed and accepted for publication by Signal Processing, Elsevier. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [http://dx.doi.org/10.1016/j.sigpro.2014.09.009].

Downloaded on 29 Mar 2024 10:35:32 SGT

Deep Transfer Learning for Classification of Time-Delayed Gaussian Networks

Iti Chaturvedi[a], Ong Yew Soon[a], R. V. Arumugam[b]

[a]*School of Computer Engineering, Nanyang Technological University, Singapore.*
[b]*Data Storage Institute, Singapore.*

Abstract

In this paper, we propose deep transfer learning for classification of Gaussian networks with time-delayed regulations. To ensure robust signaling, most real world problems from related domains have inherent alternate pathways that can be learned incrementally from a stable form of the baseline. In this paper, we leverage on this characteristic to address the challenges of complexity and scalability. The key idea is to learn high dimensional network motifs from low dimensional forms through a process of transfer learning. In contrast to previous work, we facilitate positive transfer by introducing a triangular inequality constraint, which provides a measure for the feasibility of mapping between different motif manifolds. Network motifs from different classes of Gaussian networks are used collectively to pre-train a deep neural network governed by a Lyapunov stability condition. The proposed framework is validated on time series data sampled from synthetic Gaussian networks and applied to a real world dataset for the classification of basketball games based on skill level. We observe an improvement in the range

Email addresses: iti@ntu.edu.sg (Iti Chaturvedi), asysong@ntu.edu.sg (Ong Yew Soon)

of [15-25]% in accuracy and a saving in the range of [25-600]% in computational cost on synthetic as well as realistic networks with time-delays when compared to existing state-of-the-art approaches. In addition, new insights into meaningful offensive formations in the Basketball games can be derived from the deep network.

Keywords:
Transfer Learning, Manifold, Time-delays, Variable-order, Gaussian networks, Deep Neural Networks

---

Nomenclature

n_d	=	Number of domains or classes
$x_i(\tau)$	=	Expression level of node i at time instant τ
$y(\tau)$	=	Class label for sample τ
β	=	Regression co-efficient matrix
$\boldsymbol{a}_i$	=	Parent set of a node i
$\boldsymbol{\theta}_{i,\boldsymbol{a}_i}$	=	Parameters for node i in the Bayesian network given parent set $\boldsymbol{a}_i$
N	=	Number of variables in the system
Σ	=	Covariance matrix of Gaussian node
μ	=	Mean vector of Gaussian nodes
v_i	=	Node i in the visible layer
h_j	=	Neuron j in the hidden layer
T	=	Number of data samples
r	=	The upper-bound of delay

X^s = Time series data for class s

l = Index for a hidden layer

f_l = Activation function of the hidden layer l

E = Global energy function for a DBN

W_l = Weights of the hidden layer l

α = Learning rate of a DBN

λ = Transformation factor for a motif

$\triangle\epsilon$ = Change in classification precision error

S = Gaussian network

1. Introduction

Time-delayed Gaussian networks (GN) can be derived by estimating the output measurements of each variable using a multivariate Gaussian function over the available measurements of input parent variables, such that for all admissible time-delays, the error of estimation is minimal [1], [2]. The edges in such a network represent causal interactions in dynamic systems and provide a basis for signal transduction in pathways. Signal transduction is transient; hence, the study on dynamics of the transduction is essential.

The classical time series models used ordinary differential equations to capture complex regulatory dynamics [3], [4], [5]. However, they do not work well on multivariate data with variable-order delays. In [6], the authors

introduced stochastic models; these assume an underlying hidden state of a dynamic system evolving over time. The earliest stochastic networks were Boolean, which were built using mutual information among discrete nodes [7], [8]. In order to predict the causality in networks with Gaussian or mixed nodes, Bayesian networks have been proposed [9], [10], [11], [12].

State-of-the-art Bayesian network is a directed acyclic graph where variables are present at the nodes and edges represent causal interactions among them. Conditional probabilities of nodes given parents are computed from the time series data. The variable-order Bayesian network is attained by learning transition networks between three or more structures. Now, expression of a node depends on the expression of parents from ($r > 1$) previous time points [13]. In [14], general tensor discriminant analysis was used to collectively model such higher-order networks during classification.

Accurately predicting joint multivariate probabilities with Gaussian nodes requires a lot of computational effort. For very large networks, it is customary to limit the connectivity of each node using some geometric prior [15]. This is unrealistic in practice since most real world networks require hub nodes with very high connectivity, so as to ensure robust signaling [4]. Lastly, when modeling networks with variable delays, robustness of the prediction has been established to be heavily reliant on the quality of time samples and prior knowledge. However, collecting of data from real world networks is often plagued with practical difficulties and high experimental costs.

Further, the distributed nature of most real world networks, manifests itself as intense crosstalk between pathways. In particular, states of Gaussian networks are often presumed to be stable, meaning that slight changes

in the state of a few parents does not change the expression state of the child node. This relates to the natural phenomenon of real world complex networks, where a system retains functionality in spite of turbulences by maintaining redundancy. Hence, most real world problems have inherent alternate pathways that can be learned incrementally from a stable form of the baseline. In this paper, we leverage on this characteristic and look at transfer learning to address the challenges of complexity and scalability.

1.1. Related Work

Domain adaptation aims to generalize one or more source class network(s) that are easier to learn, to augment the target network, for which data is scarce. Since data in networks of different classes are distributed differently, previous authors have collectively used data from the source(s) and the target class networks to learn shared feature representations [16]. In the case of networks, we can consider motifs, which are recurrent, or statistically significant sub-graphs shared across different classes. Deep neural networks are ideal for learning a set of shared motifs from different classes of Gaussian networks [17], [18], [19]. However, simply learning different class networks together can be detrimental when the networks are unrelated.

Transfer learning has been previously used in the Bayesian frameworks to estimate the prior covariance matrix of the target network by simply averaging the covariance matrices of previously determined structures in similar classes of networks [20], [21]. In their approach, the history of past maximum likelihood (ML) estimates for motifs can be reused by transferring to suitable new structures of higher dimensions. However, the discriminative information in the covariance matrices is often lost due to under sampling of

data. Previously, the use of normalized divergences was shown to preserve small differences between classes [22]. Transfer may be transductive which means across different dynamic systems or inductive which means transfer among related structures in the same network. The objective function tries to minimize the loss in prediction accuracy due to the transfer [23]. For example, in [24], labeled and unlabeled samples are combined with a trade-off parameter.

In document classification, instead of assuming all word probabilities are independent in the target class, transductive transfer learning was used to estimate the dependencies between words using other source classes with known labels. Since, the number of possible covariance estimates would increase exponentially with size of the vocabulary, in [20] the authors learn the covariance for only a subset of words from the source classes and combine it with the rest of the vocabulary under a semi-definite constraint. However, their approach will not be able to capture the underlying structure due to higher-order dependencies among groups of words that can occur at variable positions in a document. It will also not be able to transfer effectively among words that are synonyms and used alternatively across documents. In [25] the authors tried to address this issue through adaptive learning of higher-order dependencies in the neighbourhood of a word. Similarly, to account for lack of data in click-based methods for image ranking in web search, we can predict clicks for new images using click data from associated images. In [26], the authors achieved this through learning of manifolds for each image feature separately using a group of weights and hyper-graphs to model higher-order dependencies. Multitask learning problems such as compiler

programs use inductive transfer learning to model inter-task dependencies without the need for large amounts of training data [21]. There, again the authors approximate the prior covariance matrix for the maximum likelihood estimates from similar or related tasks.

In contrast to previous works on transfer learning, in this paper, we propose deep transfer learning for classification of Gaussian networks with time-delayed regulations. We leverage on our knowledge of real world networks, where a distributed architecture has inherent redundancies in the form of alternate pathways to ensure robustness of signalling. The core idea is to transfer maximum likelihood estimates for a small set of computed network motifs in each class and to subsequently approximate the remaining unknown motif probabilities. In this way we can reduce the computational effort necessary while learning traditional Gaussian networks. To facilitate this form of transfer, here we consider a manifold triangular inequality to transfer ML probabilities selectively from one sub-structure referred to as motif here, to another.

Motifs with ML that satisfy the threshold computed using transfer learning are then used to pre-train a deep neural network. We show that the natural layered architecture of deep networks is able to model variable delays including long delays when governed by a Lyapunov stability condition. We refer to the resulting framework as a deep transfer neural network (DTNN).

1.2. Contributions and Paper Outline

Our approach to classify networks from time series data comprises three stages: (a) Building a training database of sub-structures or motifs with maximum likelihood probabilities above a threshold. Here, we compute a

sub-set of low dimensional motifs for each class of Gaussian networks and learn the high dimensional motifs through a transfer learning algorithm with manifold inequality constraint. (b) Pre-train the weights of a deep neural network collectively with the motifs from different classes of Gaussian networks learned in the first step governed by the Lyapunov stability condition. This is followed by training using time series data from the different classes of Gaussian networks. Lastly, the deep neural network tries to minimize the classification error on the training motifs and time samples using the known class labels. (c) Classify the testing time samples from different class networks with time delays using the trained deep transfer neural network. In addition, the structural formations of the hidden neurons in the layers provide new insights into the GNs.

The following is a summary of the significance and contributions of the research work presented in this paper:

- We introduce a deep transfer neural network (DTNN) capable of classifying dynamic systems. From our knowledge, no previous work has considered deep transfer learning to model time-delays in dynamic systems.

- To facilitate positive transfer of learned maximum likelihood from source motif to the target motif with a manifold triangular inequality.

- We consider both inductive learning of related tasks inside a large system as well as transductive transfer learning of target motifs from the source motifs at a much lower computational cost.

- The DTNN is collectively pre-trained using motifs with ML probabili-

ties that satisfy the heuristically defined threshold. Due to the sharing of data in the layers of a deep network, prediction is feasible even with few time samples.

Validation of the method is then performed using synthetic and real world datasets. Taking the cue from the approach described in [27], synthetic time series data were generated from nodes following a Gaussian distribution. Comparison is then conducted with previously proposed algorithms for classification. The results obtained showed that the proposed approach outperforms all the existing baseline methods considered on the synthetic Gaussian networks and is able to classify different Gaussian networks with time-delays with high accuracy. We also performed experiments to measure the trade-off between the number of nodes in the Gaussian network and the number of training time samples available. We then conclude that the DTNN requires much fewer time samples than existing techniques.

Subsequently, we also applied the DTNN on real world video data collected from basketball matches. The behaviours, particularly the trajectories of five players in basketball matches are considered for classifying a match or game as a *beginner* game or a *skilled* game. From our study, the offensive formations derived for *beginner* and *skilled* games were found to be statistically different.

The organization of the paper is as follows: Section 2 provides the preliminary concepts necessary to understand the present work. In section 3, we introduce the proposed deep transfer learning and describe the algorithm for learning the weights of a DTNN framework. Lastly, in section 4, we validate our method on synthetic and real datasets.

2. Preliminaries

In this section, we briefly review the theoretical concepts necessary to comprehend our proposed algorithm. We begin with a description of regression models. This is followed by a review of parameter learning in dynamic Bayesian networks (BN). In contrast to BN, we show that weights in the deep neural network are learned by maximizing a global energy function.

Notations: Consider a set of n_d Gaussian networks (GN) to be classified. Each GN is a set of N nodes and time series of observations gathered over T time points for all the nodes. Nodes can take real values from a multivariate distribution determined by the parent set. In this paper, we presume that samples are collected over T equally spaced time points. Let the dataset of samples be $X = \{x_i(\tau)\}_{N\times T}$, where $x_i(\tau)$ represents the sample value of the i $^{\text{th}}$ random variable at time point τ. Lastly, let $\boldsymbol{a}_i$ be the set of parent variables regulating variable i.

2.1. Multivariate Autoregression

A multivariate autoregressive (MVAR) model provides a stochastic framework for learning GN among a set of random variables [28]. We denote the upper bound of delay in the Gaussian network with r. An r-order MVAR classifier model with delays is given by:

$$y(\tau) = \frac{1}{1 + exp(\sum_{\tau'=1}^{r} \beta^{\tau} \mathbf{x}(\tau - \tau') + \boldsymbol{\varepsilon}(\tau))}, \tag{1}$$

where $0 < y(\tau) < 1$, and the discretized form $\hat{y}(\tau) \in \{1, 2, \ldots, n_d\}$ is the Gaussian network class, $\beta^{\tau} = \{\beta^{\tau}_{i,j}\}_{N\times N}$ denotes the matrix of regression coefficients corresponding to a delay of τ time points, and $\boldsymbol{\varepsilon}(\tau) = (\varepsilon_i(\tau))_{i=1}^{N}$

denotes i.i.d. zero mean additive Gaussian noise. The regression coefficient $\beta_{i,j}^t$ when $i \neq j$, represents the interaction between the two nodes: i and j [29]. The structure of the Gaussian network is obtained via forcing small coefficients to zero. However, with increasing size and cross talk in the networks, it is more efficient to represent GN as a graphical model such as the Bayesian network.

2.2. Dynamic Bayesian Networks

A Bayesian network is a graphical model that represents a joint multivariate probability distribution for a set of random variables [28]. It is a directed acyclic graph having a structure S and a set of parameters $\boldsymbol{\theta}$ that represent the strengths of connections by conditional probabilities. The BN decomposes the likelihood of node expressions into a product of conditional probabilities by assuming independence of non-descendant nodes, given their parents.

$$p(X|S,\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{a}_i, \theta_{i,\boldsymbol{a}_i}), \tag{2}$$

where $p(\mathbf{x}_i|\boldsymbol{a}_i, \theta_{i,\boldsymbol{a}_i})$ denotes the conditional probability of node expression $\mathbf{x}_i$ given its parent node expressions $\boldsymbol{a}_i$, and $\theta_{i,\boldsymbol{a}_i}$ denotes the ML estimate of the conditional probabilities. Fig. 1 (a) illustrates the Bayesian network for a multivariate system with five nodes. Each node is a variable in the state-space of the system that can be observed or measured. The connections represent causal dependencies within a single time instant. The observed state vector of variable i is denoted as $\mathbf{x}_i$ and the regulation or conditional probability of variable i given variable j is $p(\mathbf{x}_i|\mathbf{x}_j)$. Fig. 1 (b) shows the Gaussian network that describes a team of five Basketball players. The black nodes denote

players of the defending team while the white nodes denote players of the offending team [30].

The optimal structure S^* is obtained by maximizing the posterior probability of S given the data X. From Bayes theorem, the optimal structure S^* is given by

$$S^* = \arg\max_S p(S|X) = \arg\max_S p(S)p(X|S), \tag{3}$$

where $p(S)$ is the probability of the network structure and $p(X|S)$ is the likelihood of the expression data given the network structure.

Given the set of conditional distributions with parameters $\boldsymbol{\theta} = \{\theta_{i,\boldsymbol{a}_i}\}_{i=1}^N$, the likelihood of the data is given by

$$p(X|S) = \int p(X|S,\boldsymbol{\theta})p(\boldsymbol{\theta}|S)d\theta, \tag{4}$$

To find the likelihood in (4), and to obtain the optimal structure as in (3), the data are pre-assumed to be either Gaussian or multinomial [31]. Multinomial models are scalable, however, when the data are continuous, their accuracy is low [32], [6]. Gaussian BN assumes that the nodes are multivariate Gaussian. That is, node expression can be described with mean μ and covariance matrix Σ of size $N \times N$ [33]. The joint probability of the network can be the product of a set of conditional probability distributions given by:

$$p(\mathbf{x}_i|\boldsymbol{a}_i) = \theta_{i,\boldsymbol{a}_i} \sim \mathcal{N}\Big(\mu_i + \textstyle\sum_{j\in\boldsymbol{a}_i}(\mathbf{x}_j - \mu_j)\beta,\ \Sigma_i^{'}\Big), \tag{5}$$

where $\Sigma_i^{'} = \Sigma_i - \Sigma_{i,\boldsymbol{a}_i}\Sigma_{\boldsymbol{a}_i}^{-1}\Sigma_{i,\boldsymbol{a}_i}^T$ and β denotes the regression coefficient matrix, $\Sigma_i^{'}$ is the conditional variance of $\mathbf{x}_i$ given its parent set $\boldsymbol{a}_i$, $\Sigma_{i,\boldsymbol{a}_i}$ is the

covariance between observations of $\mathbf{x}_i$ and the variables in $\boldsymbol{a}_i$, and $\Sigma_{\boldsymbol{a}_i}$ is the covariance matrix of $\boldsymbol{a}_i$.

The acyclic condition of BN does not allow self- and feedback- regulations of nodes, which are essential characteristics of GN [34],[35]. Therefore, dynamic Bayesian networks (DBN) have recently become popular in building GN from time series data mainly due to their ability to model causal interactions as well as feedback regulations [36]. The dynamic form of a Bayesian network is shown in Fig. 1 (c). It assumes a first-order Markov assumption, that parents from a previous time-instant regulate nodes and learns the transition network between them. A first-order DBN is defined by a transition network of interactions between a pair of structures connecting nodes at time instants τ and $\tau + 1$. In time instant $\tau + 1$, the parents of nodes are those specified in the time instant τ. Similarly, the structure of an r-order DBN is represented by a connectivity structure comprising $(r + 1)$ consecutive time points and N nodes, or a graph of $(r + 1) \times N$ nodes.

It is easy to visualize that in a variable-order DBN, the number of nodes increases exponentially at very high-orders, making it computationally intractable. To circumvent this issue, skip-edges are used. Skip-edges appear as jump ahead connections, providing a shorter path for propagation of the conditional probability. The skip-edge probability is decomposed into a sum of terms for consecutive pairs of nodes with first-order dependencies and the most likely cascade of interactions can be found using the Viterbi algorithm [37]. In the next section, we describe the use of deep neural networks for learning skip-edges efficiently.

2.3. Gaussian Deep Neural Networks

Deep neural network (DNN) consists of multiple layers of latent or hidden neurons, such that each layer takes the form of a simpler model such as the restricted Boltzmann machines (RBM). RBM is a bipartite graph comprising two layers of neurons: a visible and a hidden layer; it is restricted so that the connections among nodes in the same layer are not permitted. Fig. 2 (a) shows a conventional RBM with two hidden neurons and three nodes in the visible layer, no edges are available between neurons in the same layer.

To model Gaussian networks with time-delays, the visible layer of the DNN consists of N nodes, where N is the number of variables in the time-delayed network. The hidden layers, on the other hand, can have an arbitrary number of neurons. The visible nodes now correspond to the Gaussian Bayesian network described in the previous section capable of representing time-delays. For example, Fig. 2 (b) shows the state space of an RBM for Gaussian networks with time delays. It has a single hidden neuron and three visible nodes. The visible nodes correspond to a Gaussian Bayesian network with three variables. Here the directed edges are estimated by training the weights of the connections between visible nodes and hidden neuron from the training time series data.

To train such a multi-layer system, we must compute the gradient of the total energy function E with respect to weights in all the layers. The output $\mathbf{x}_l = f(\mathbf{x}_{l-1}, W_l)$ of each layer l in a DNN is a function of the inputs $\mathbf{x}_{l-1}$ from the previous layer as well as the layer weight W_l.

To learn these weights and maximize the global energy function, the approximate maximum likelihood contrastive divergence (CD) approach can

be used. This method employs each training sample to initialize the visible layer. Next, it uses the Gibbs sampling algorithm to update the hidden layer and then reconstruct the visible layer consecutively, until convergence [38].

As an example, here we use a logistic regression model to learn the binary hidden neurons and each visible unit is assumed to be a sample from a normal distribution [18]. The continuous state $\hat{h}_j$ of the hidden neuron j, with bias b_j, is a weighted sum over all continuous visible nodes $\boldsymbol{v}$ and is given by:

$$\hat{h}_j = b_j + \sum_i v_i w_{ij}, \tag{6}$$

where w_{ij} is the connection weight to hidden neuron j from visible node v_i (see Fig. 2 (a)). The binary state h_j of the hidden neuron can be defined by a sigmoid activation function:

$$h_j = \frac{1}{1 + e^{-\hat{h}_j}}. \tag{7}$$

Similarly, in the next iteration, the binary state of each visible node is reconstructed and labelled as $\boldsymbol{v}_{recon}$. Here, we determine the value to the visible node i, with bias c_i, as a random sample from the normal distribution where the mean is a weighted sum over all binary hidden neurons and is given by:

$$\hat{v}_i = c_i + \sum_j h_i w_{ij}, \tag{8}$$

where w_{ij} is the connection weight to hidden neuron j from visible node v_i, (see Fig. 2 (a)). The continuous state v_i is a random sample from $\mathcal{N}(\hat{v}_i, \sigma)$, where σ is the variance of all visible nodes. Lastly, the weights are updated as the difference between the original and reconstructed visible layer using:

$$\triangle w_{ij} = \alpha(< v_i h_j >_{data} - < v_i h_j >_{recon}), \tag{9}$$

where α is the learning rate and $< v_i h_j >$ is the expected frequency with which visible unit i and hidden unit j are active together when the visible vectors are sampled from the training set and the hidden units are determined by (6). Finally, the energy of a DNN can be determined in the final layer using $\mathrm{E} = -\sum_{i,j} v_i h_j w_{ij}$. When a DNN is trained with data from different GN, negative transfer might occur due to variable-order time-delays. Hence, in this paper we propose a deep transfer neural network to classify GN with time delays where positive transfer is facilitated through pre-training.

3. Deep Transfer Learning Framework

The stability of a real world system depends on the ability of the Gaussian network to switch among different modes in response to environmental turbulences. Different alternate Gaussian networks are active in related networks and at different stages of the response. In this section, we introduce a novel deep transfer neural network that leverage's on such natural redundancies in alternate networks to classify GN with time-delays. We achieve this through inductive as well as transductive transfer of motifs from source to target manifolds.

When a deep neural network is pre-trained with motifs learned through transfer, the resulting model is referred to as a deep transfer neural network (DTNN). In the context of Bayesian networks, a motif has the following definition:

Algorithm 1 Deep Transfer Learning of Gaussian Networks

Input 1: Training time series $\boldsymbol{X} = \{X^1, X^2, \ldots, X^{n_d}\}$ for all n_d networks.
Input 2: ML parameters of motifs $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^{n_d}\}$ learned via transfer in **Algorithm 2**.
Outputs: Gaussian network class label for each test sample
$X^0 = \{\theta_{i,\boldsymbol{a}_i} \geq \boldsymbol{\gamma}\}, \forall \theta_{i,\boldsymbol{a}_i} \in \boldsymbol{\Theta}$
Update: $\boldsymbol{X} = \{X^0 \cup \boldsymbol{X}\}$
Construct the visible layer as a vector of N expression values
Construct a minimal network with single hidden neuron
Construct the output layer with n_d neurons to classify all GNs
repeat
    for $s = 0$ to n_d **do**
        for $t = 1$ to $|X^s|$ **do**
            Initialize the visible layer with t^{th} training sample in X^s
            Update W_l using CD given by (9) $\forall l$
        end for
    end for
    Compute change in classification precision error $\triangle\epsilon$ on training data $\boldsymbol{X}$
    if $\triangle\epsilon$ is significant **then**
        Add another hidden neuron
        Initialize weights of the layer to $\{w_{i,j}\}_{N \times N} = I$
    else
        Add another layer with a single neuron
    end if
until Adding a layer does not change precision error
Fine-tune weights using known Gaussian network class label of training samples
Classify test time samples to Gaussian networks using trained DTNN

Definition 1. *Network motifs are recurrent or statistically significant subgraphs. In our present study, a motif refers to a single node and its parent nodes. All dependencies are first-order Markovian and are predicted using (5) from time series data. The ML probability of a motif $\theta_{i,\boldsymbol{a}_i}$ in a GN is the conditional probability of the child node i given parent set $\boldsymbol{a}_i$.*

To mitigate the effect of negative transfer, we consider a manifold triangular inequality that provides a measure for the feasibility of transfer from the source to the target manifold. Since, time series for GN can be of variable lengths, here we consider Kullback-Leibler divergence to transfer ML probabilities of motifs from source to target network. **Algorithm 1** details the deep transfer neural network framework for classification of GN with time delays. Pre-training is done using motifs with ML that satisfy a threshold, screened and computed through the transfer learning algorithm described in the next sub-section.

3.1. Building a training database of motifs through transfer with manifold inequality constraint

In [39], the author has provided a sufficient condition for mapping between two differentiable manifolds. They show that, one manifold is a transformation of the other as long as the triangular inequality is satisfied. Here, we extend the notion to similar manifolds of the source and the target motifs. In particular, we consider the sectional curvature $c > 0$ at an arbitrary point p in the source manifold, then the following condition should hold for arbitrary points q and r in the target manifold for feasibility of transfer:

$$d(p,q) + d(q,r) + d(r,p) \leq 2\pi/\sqrt{c}\ ,\ d(q,r) \leq \pi/\sqrt{c}, \tag{10}$$

where d is the distance in z-dimensional space and z is the dimensionality of the target motif. The first inequality in (10) ensures that the change from source manifold to target manifold does not alter the distances between nodes of the motif, while the second inequality constrains the curvature of the second manifold so that the diameter of the local sphere in the target manifold does not exceed $\pi/\sqrt{c}$ [39]. It can be reasoned that the area of the local sphere in both manifolds is similar. Fig. 3 illustrates the source motif and the target motif manifolds. We can determine if the manifolds are mappable using the inequality (10). The inequality constrains the curvature c at point p in the source manifold and the diameter of sphere passing through p and two other arbitrary points q and r in the target manifold. Details of the proof are provided in [39].

Our proposed transfer learning approach comprises two steps. First, we find a low dimensional representation for both the source and the target motifs through principal component analysis such that the new motifs are of equal dimensions. As a second step, we find the transformation factor between the new motifs of equal dimensions. We model the manifold alignment function as a linear mapping $f : \theta^s_{i,\boldsymbol{a}_i} \rightarrow \theta^t_{i,\boldsymbol{a}_i}$ from the source motif to the target motif.

$$f(\theta^s_{i,\boldsymbol{a}_i}) = A * \theta^s_{i,\boldsymbol{a}_i}, \tag{11}$$

where $A \in R^{z \times z}$ is a transformation matrix with z as the common dimension of the manifolds.

Rough alignment can be used when the number of data points is different in the target and source manifolds. Here we assume that both datasets are Gaussian distributed and we minimize the distance between the two Gaussians $\theta^s_{i,\boldsymbol{a}_i}$ and $\theta^t_{i,\boldsymbol{a}_i}$ known as Kullback-Leibler divergence. As proven in [40], the corresponding transformation matrix is given by

$$A = \wedge_t^{1/2} \wedge_s^{-1/2}, \tag{12}$$

where $\wedge_s$ and $\wedge_t$ are diagonal matrices with eigenvalues of Σ_{ss} and Σ_{tt}, respectively. The first eigenvalue λ corresponding to child node in matrix A is used to scale the ML probability of source motif to that of the target motif.

In very large networks, the computational bottleneck lies in the estimation of the likelihood of nodes given their parents. Hence, we use the time series data to only extract a sub-set of probabilities for each node given a possible parent set. We refer to these sub-structures as motifs. The remaining sub-structures are then inferred using transfer learning at very low computational cost governed by the triangular inequality. In the next sub section, we describe the transfer learning of high dimensional motifs within a single large network as well as from other source networks.

3.2. Transfer Learning of Motifs

An outline of the procedure for the transfer learning of motifs is summarized in **Algorithm 2**. The algorithm takes as input the training time series $\boldsymbol{X} = \{X^s\}_{s=1}^{n_d}$ for all n_d classes of GNs and learns the corresponding motifs $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^s\}_{s=1}^{n_d}$. This requires two steps: (i) First we compute a subset of low dimensional motifs $\boldsymbol{\theta}^s$ for all classes of Gaussian networks using (5) such that the number of parents for any node $|\boldsymbol{a}_i| \leq d$. Motifs with ML probability

above threshold γ are used to construct a database $\boldsymbol{M}$ of candidate source motifs. (ii) Next, we use transfer learning to incrementally learn high dimensional motifs in all classes of GN from the candidate motifs in $\boldsymbol{M}$. To do this, we create a new target motif $\theta_{i,\boldsymbol{a}_i}^{new}$ by perturbing a node(s) in a candidate source motif $\theta_{i,\boldsymbol{a}_i}$ with a node(s) from the class t of the target motif. Next, the feasibility of mapping between motif $\theta_{i,\boldsymbol{a}_i}^{new}$ and motif $\theta_{i,\boldsymbol{a}_i}$ are determined using (10). If the triangular inequality holds, then the transformation factor λ is computed using (12), and the new motif probability is the product of λ and $\theta_{i,\boldsymbol{a}_i}$. The new motif and its probability is then added to the parameter set $\boldsymbol{\theta}^t$ of the target motif class. This process is repeated for all candidate source motifs in $\boldsymbol{M}$ and all possible target motifs in class t until the termination condition is reached. Here, the algorithm terminates when all possible motifs have been learned or when the computational time available has been exhausted. In this way, we can save on huge computational cost associated with learning of high dimensional Gaussian motifs. This process is repeated for each Gaussian network as the target class and all the Gaussian networks being the source class. Interestingly, in the case where the class of target motif and source motif is common the algorithm corresponds to inductive transfer learning.

3.3. Training a deep neural network with motifs learned through transfer

To begin, the training time samples set $\boldsymbol{X} = \{X^1, X^2, \ldots, X^{n_d}\}$ is updated to include motifs learnt through transfer (see **Algorithm 2**). Here again, we only consider motifs with ML probability above γ to construct a time series X^0. Each training motif shall serve as a new training sample which is a vector of N expression values. Nodes occurring in a motif are set

Algorithm 2 Transfer Learning of Motifs

Input : Training and testing time series $\boldsymbol{X} = \{X^1, X^2, \ldots, X^{n_d}\}$ for all n_d class networks
Output : ML parameters of motifs $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^{n_d}\}$
for $s = 1$ to n_d **do**
    $\boldsymbol{\theta}^s = \{\theta_{i,\boldsymbol{a}_i} = p(\mathbf{x}_i|\boldsymbol{a}_i), |\boldsymbol{a}_i| \leq d\} \forall i, \forall \boldsymbol{a}_i$ using (5) and time series X^s.
    $\boldsymbol{M} = \{\theta_{i,\boldsymbol{a}_i} \geq \gamma\}, \forall \theta_{i,\boldsymbol{a}_i} \in \boldsymbol{\theta}^s$
    for $t = 1$ to n_d **do**
        repeat
            for $m = 1$ to $|\boldsymbol{M}|$ **do**
                Obtain new parent set $\boldsymbol{a}_i^{new}$ by perturbing node(s) in $\theta^m_{i,\boldsymbol{a}_i}$ with a node(s) in target class network
                if $\{i, \boldsymbol{a}_i^{new}\}$ and $\{i, \boldsymbol{a}_i^m\}$ satisfy inequality (10) **then**
                    $\lambda =$ solve (12) using $\{X^t, X^s\}$
                    ML probability $\theta^{new}_{i,\boldsymbol{a}_i} = \lambda * \theta^m_{i,\boldsymbol{a}_i}$
                    Update : $\boldsymbol{\theta}^t = \{\boldsymbol{\theta}^t \cup \theta^{new}_{i,\boldsymbol{a}_i}\}$
                end if
            end for
        until Termination condition is reached
    end for
end for

to one and expressions of all other nodes were set to zero. Motifs of different classes of Gaussian networks are used collectively to pre-train a deep neural network governed by the Lyapunov stability condition. Since the motifs have already been validated for feasibility of transfer across different classes of Gaussian networks, the likelihood that the shared motifs learned in the first hidden layer would facilitate positive than negative transfer is higher.

Interestingly, a DTNN is capable of learning from sub-structures; hence we can compute as many motifs as the computational time permits. While, it is desirable to compute all possible motifs, a DTNN can learn well even from a small sub-set of motifs. We begin with a minimal configuration of a N visible nodes, a single hidden neuron and n_d output neurons to classify the Gaussian networks (see **Algorithm 1**). To determine the optimal number of hidden neurons in a single layer as well as the number of layers, we compute the change in classification precision error $\triangle\epsilon$ of the DTNN on training samples. If there is a significant change in the error $\triangle\epsilon$, a new hidden neuron is added. The layer weights are then relearned and the classification precision error is recomputed. The process is repeated until no significant change in precision error is observed with the addition of hidden neurons. When this happens, a new hidden layer is added. The above progresses iteratively until further addition of layers of hidden neurons does not change the classification precision error significantly, and the optimal configuration is arrived.

3.4. Initialization of weights and classification of different Gaussian networks with time-delays

The contrastive divergence approach will sample motifs with high-frequency into the upper layers, resulting in the formation of class specific sub-graphs

at hidden neurons in the first layer, class specific bigger graphs at hidden neurons in second hidden layer and so on. However, for Lyapunov stability it is desirable that the covariance matrix of the complete network is close to positive semi-definite. Hence, we initialize weights in each hidden layer to a diagonal matrix which is close to a feasible solution.

The final output layer has n_d hidden neurons so that it can classify a sample input vector among n_d classes of Gaussian networks. After training, the weights are fine-tuned using back propagation algorithm with the known class label of the training motifs and time samples. The trained DTNN is then used to classify test time samples from different class of GNs.

3.5. Complexity Analysis

Computing the likelihood of a motif using (5) requires computing the second-order derivative of the covariance matrix (Hessian), whose complexity is known to be at least quadratic in the number of dimensions [41]. For a network of N nodes, the total number of possible motifs can be derived as $\sum_{i=1}^{N} {}^{N}C_i$

Here, since we only compute the likelihood of partial motifs (i. e., the low dimensional ones with up to a maximum of K nodes, where $K <<< N$) to approximate all the likelihoods, the total number of motifs to be computed becomes $\sum_{i=1}^{K} {}^{n}C_i$. Thus, the cost of computation is reduced from $O(\sum_{i=1}^{N} {}^{N}C_i.i^2)$ to $O(\sum_{i=1}^{K} {}^{N}C_i.i^2 + \delta)$ where $O(i^2)$ is the cost to compute a single motif of i nodes and δ is the cost of approximation, which is negligible in comparison.

4. Experiments

In this section, the proposed DTNN was applied to two dynamic Gaussian network classification problems in order to assess its efficacy. The first dataset was generated synthetically from networks of size $N \in (10/20)$ with variable delays of regulation up to three time points. As considered in literature, we refer to the 10 node Gaussian network as small, while the 20 node Gaussian network is an example of a large dynamic system. Following the approach described in [27], random delays were associated with edges before sampling time series data from a multivariate Gaussian distribution. The other dataset was real world time-series collected from Basketball games.

Performance measures such as precision[1], recall[2], and F-measure[3] were evaluated using true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

Before we discuss the performance of the DTNN on the datasets individually, we define the parameter settings common to all experiments:

- The likelihood of motifs are extracted from the time series data using (5) and the transfer learning algorithm.
- Top 20% of motifs are used to determine the threshold γ. Motifs above this threshold γ are used to construct the training samples for the deep transfer neural network.
- The learning rate for deep neural network was set to $\alpha = 0.001$ and

[1]Precision $= TP/(TP + FP)$
[2]Recall $= TP/(TP + FN)$
[3]F-measure $=2\ \frac{Precision \times Recall}{Precision + Recall}$

the maximum number of parents for low-dimensional motifs was set to $d = 3$.

- The optimal number of hidden neurons and the number of layers are determined heuristically by computing the change in classification precision error on the training samples.

4.1. Deep Transfer Learning of Gaussian Networks

In this section, we discuss the process to derive the topologies of the target synthetic Gaussian networks that mimic real world problems. First, we briefly explain how time series datasets are sampled from the synthetic Gaussian networks. Results on varying network sizes, number of time samples and computation time are discussed. Lastly, we verify the performances of our method against several state-of-the-art baseline algorithms.

4.1.1. Generating Synthetic Time Series

In this study, the time series dataset of expressions was generated from various Gaussian networks with varying number of nodes. The nodes were assumed to take real values from a multivariate Gaussian distribution. Similar to the approach described in [27], two different class Gaussian networks were considered with the mean of each node set to $(+5/-5)$ and co-variance between nodes was set to $(1/2)$ in the first time stamp. For all remaining time stamps, node expressions were sampled from MVAR, using (1).

4.1.2. Comparison with State-of-the-art Baseline Algorithms

We compared our method with several state-of-the-art baselines for classification including MVAR [29], SVM [2] [42] and DNN [20]. Training time series datasets of length (100/30) were generated from two different Gaussian networks of size (10/20) with up to third-order time delays. Table 1 tabulates the results including TP, precision, recall, and F-measure for classifying the synthetic Gaussian networks of (10/20) nodes when the number of training time samples available is (100/30). All the models are evaluated on a balanced dataset of 20 test samples (10 from each class). Here, we have reported the weighted average of classification precision, recall and F-measure over both the classes. Further, in order to demonstrate the robustness of our method we have reported mean and variance over five different test datasets for each model.

Fig. 4 shows the results on the predicted synthetic Gaussian networks. In Fig. 4 (a), comparison of the F-measure for the DTNN with the baselines on the smaller Gaussian network of 10 nodes when (100/30) training time samples are available is presented. It is shown that in contrast to the baselines, DTNN exhibits high classification accuracy even when only 30 time samples are used. In Fig. 4 (b) the F-measures for the larger Gaussian network of 20 nodes is depicted. Here, DTNN outperformed the baseline SVM by over 20% even when 100 time samples were used. When using a shorter time-series of length 30, an even bigger margin of 30% improvements in performance was observed.

[2]Support Vector Machines (SVM)

The percentage of improvements in F-measure as attained by the DTNN over the baseline algorithms for Gaussian networks of size 10 and 20 when 100 training time samples are available are then summarized in Fig. 4 (c). In particular, the percentage of improvement in F-measure (PI_M)[5] as attained by DTNN over MVAR, SVM and DNN for Gaussian networks of size 10 and 20, are reported. It can be seen that on the larger Gaussian network of 20 nodes, our method has almost 25% improvements in accuracy over the baselines. For the Gaussian network of 10 nodes, our method exhibited up-to 15% improvements in accuracy as compared to the baselines.

4.1.3. Computation Time

Table 2 tabulates the wall-clock time incurred to compute the maximum likelihood probabilities for Gaussian networks of size $n \in \{10/20\}$ nodes, with and without the use of transfer learning. The second column of Table 2 shows the cost of computing all the 3 node and 4 node motifs. In the third column, we show the cost of computing only the low dimensional motifs of 3 nodes and approximating the high dimensional motifs of 4 nodes using transfer learning. It is found that at least 25% savings can be attained when performing a transfer learning on the 10 nodes Gaussian network. A much higher savings of over 600% was observed on the larger Gaussian network of 20 nodes.

4.2. Real World Dataset

We also applied our proposed algorithm to a video dataset involving a small Gaussian network to classify the behaviour of players in a basketball

[5] $PI_M = (F_{DTNN} - F_M)/F_{DTNN}$

game as a *beginner* game or a *skilled* game using the player trajectories captured through a camera [43]. Our method exhibited notable results on this realistic problem. On a real world dataset, the true target Gaussian network is unknown; hence, we use the classification accuracy on the testing set as a measure of precision.

4.2.1. Basketball Game

In basketball, it is possible to determine the offensive and defensive teams using player behavioural tracking [30]. Previously, dynamic BN was used to classify the entire basketball game as a *beginner* game or a *skilled* game [43]. They showed that differences in likelihoods of winning team from the losing team were statistically different for *beginner* and *skilled* games. However, the variance of their prediction was very large. Here, we show that a DTNN is able to provide a much better classification measure.

The trajectories of five players during a basketball game can be represented as a GN network with five nodes. The behaviour of the players or the time-delayed regulations between the players, such as passing of the ball, can be modelled as variable-order edges in the Gaussian network. 15 games of variable durations of each type were used to train the DTNN. The behavioural data of beginner players and skilled players in all 15 games is used to compute the likelihood of low dimensional motifs using (5) and high dimensional motifs using transfer learning methodology. Subsequently, the top 20% of both low dimensional and high dimensional motifs then form the training samples for constructing the DTNN.

The optimal configuration determined by our algorithm was a two layer DTNN with hidden layers given by [5, 10, 2]. The visible layer has 5 nodes

and the first hidden layer had 10 neurons. No significant improvement was observed with additional layers. We compared the performance of the DTNN using precision, recall, and F-measure, with the baselines on 20 testing time samples from both game types. The DTNN was able to correctly classify all of the test samples; however the other methods showed almost half the accuracy compared to DTNN.

As an illustration, Fig. 5 depicts the offensive behaviour and formations involving a team of five players in a game of Basketball. The basket is shown at the mid-point of the court width. The figurines correspond to five players in a team. The formations in Fig. 5 (a) to (e) correspond to the hidden neurons, where weights of the connection to the *skilled* class in the final layer are higher than the *beginner* class neuron. The formations in Fig. 5 (f) to (j), on the other hand, correspond to hidden neurons where weights of the connection to the *beginner* class in final layer are higher than that of the *skilled* game neuron. Formations in *skilled* games show a low (< 0.5) Pearson correlation measure since the players are spread across the Basketball court. Contrastingly, the formations of *beginner* games show a high (> 0.5) Pearson correlation measure since the players are observed to be herding near the basket.

5. Conclusions

In this paper, we have proposed a deep transfer neural network to classify Gaussian networks with time-delays. Our simulation and experimental study showed that the proposed DTNN outperformed the baseline methods in classification accuracy. The method also required fewer time samples to classify

high-dimensional Gaussian networks. We observed an improvement in the range of [15-25]% in accuracy and a saving in the range of [25-600]% in computational cost on synthetic as well as realistic networks with time-delays, when compared to existing state-of-the-art approaches.

Classification of temporal processes using deep neural networks is a challenging task due to the multi-modal nature of the underlying distribution. To address this, we pre-trained the deep neural network with high ML motifs from different classes of Gaussian networks with time-delays computed using dynamic Bayesian fitness function.

Since, only a small set of low dimensional ML probabilities are computed while the remaining are approximated by a transfer learning algorithm, we were able to save on tremendous computational time and achieve improved network scalability. In order to determine the feasibility of transfer between low dimensional source motifs and the high dimensional target motifs, we considered the triangular manifold inequality.

Further, the DTNN learns the shared representation of low dimensional motifs for all classes of Gaussian networks at the hidden neurons in the first layer. The higher layers combine these motifs to form bigger structures that may occur in one or more class of Gaussian networks. Finally, in the last layer each node corresponds to the complete Gaussian network of a single class. In this way, there is data sharing across sub-structures and many fewer time samples are needed for prediction. To ensure stochastic stability of the complete Gaussian network, the learning of weights is governed by a Lyapunov stability condition. Last but not the least, it can be observed that the higher layers in the DTNN can learn long delays by cascading low-order

time-delays in the layers below.

In the future the DTNN will be used to classify and bring insight into the time-varying data access patterns of software programs in large-scale cloud data centers. Related programs will have several common sub-modules that may be learned through transfer.

6. Acknowledgement

This work is partially supported by the ASTAR Thematic Strategic Research Programme (TSRP) Grant No. 1121720013 and the Center for Computational Intelligence (C2I) at NTU.

References

[1] N. Friedman, I. Nachman, Gaussian process networks, in: Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00), 2000, pp. 211–219.

[2] A. Schwaighofer, M. Dejori, V. Tresp, M. Stetter, Structure learning with nonparametric decomposable models (2007).

[3] J. D. Hamilton, Time Series Analysis, 1994.

[4] H. Liu, J. Lafferty, L. Wasserman, The nonparanormal: Semiparametric estimation of high dimensional undirected graphs, J. Mach. Learn. Res. 10 (2009) 2295–2328.

[5] Z. Wu, P. Shi, H. Su, J. Chu, Delay-dependent state estimation for discrete markovian jump neural networks with time-varying delay, Asian Journal of Control 13 (6) (2011) 914–924.

[6] D. Heckerman, D. Geiger, D. M. Chickering, Learning bayesian networks: The combination of knowledge and statistical data, Machine Learning 20 (3) (1995) 197–243.

[7] P. Li, C. Zhang, E. J. Perkins, P. Gong, Y. Deng, Comparison of probabilistic boolean network and dynamic bayesian network approaches for inferring gene regulatorynetworks, BMC Bioinformatics 8 (2007) S13–S20.

[8] T. Akutsu, S. Miyano, S. Kuhara, Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function, Journal of Computational Biology 7 (3-4) (2000) 331–343.

[9] J. Pearl, Probabilistic reasoning in intelligent systems : networks of plausible inference, The Morgan Kaufmann series in representation and reasoning., : Morgan Kaufmann Publishers, 1988.

[10] S. Imoto, S. Kim, T. Goto, S. Miyano, S. Aburatani, K. Tashiro, S. Kuhara, Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network, J Bioinform Comput Biol 1 (2) (2003) 231–52.

[11] N. Nariai, S. Kim, S. Imoto, S. Miyano, Using protein-protein interactions for refining gene networks estimated from microarray data by bayesian networks, Pac Symp Biocomput (2004) 336–47.

[12] K. Ota, T. Yamada, Y. Yamanishi, S. Goto, M. Kanehisa, Compre-

hensive analysis of delay in transcriptional regulation using expression profiles, Genome Informatics 14 (2003) 302–303.

[13] Z. Xing, W. Dan, Modeling multiple time units delayed gene regulatory network using dynamic bayesian network, in: Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on, 2006, pp. 190–195.

[14] D. Tao, X. Li, X. Wu, S. Maybank, General tensor discriminant analysis and gabor features for gait recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (10) (2007) 1700–1715.

[15] N. Meinshausen, P. Buhlmann, High-dimensional graphs and variable selection with the Lasso, The Annals of Statistics 34 (3) (2006) 1436–1462.

[16] M. Chen, Z. Xu, K. Weinberger, F. Sha, Marginalized denoising autoencoders for domain adaptation, in: Proceedings of the 29th International Conference on Machine Learning (ICML-12), ICML '12, 2012, pp. 767–774.

[17] H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, 2009, pp. 609–616.

[18] G. W. Taylor, G. E. Hinton, S. T. Roweis, Modeling human motion using binary latent variables, in: B. Schölkopf, J. Platt, T. Hoffman

(Eds.), Advances in Neural Information Processing Systems 19, MIT Press, Cambridge, MA, 2007, pp. 1345–1352.

[19] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.

[20] R. Raina, A. Y. Ng, D. Koller, Constructing informative priors using transfer learning., in: ICML, 2006, pp. 713–720.

[21] E. V. Bonilla, K. M. Chai, C. Williams, Multi-task gaussian process prediction, in: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems 20, 2008, pp. 153–160.

[22] D. Tao, X. Li, X. Wu, S. Maybank, Geometric mean for subspace selection, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (2) (2009) 260–274. doi:10.1109/TPAMI.2008.70.

[23] C.-W. Seah, I. W. Tsang, Y.-S. Ong, Transductive ordinal regression., IEEE Trans. Neural Netw. Learning Syst. 23 (7) (2012) 1074–1086.

[24] J. Yu, J. Cheng, D. Tao, Interactive cartoon reusing by transfer learning, Signal Processing 92 (9) (2012) 2147 – 2158.

[25] J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification, Image Processing, IEEE Transactions on 21 (7) (2012) 3262–3272.

[26] J. Yu, Y. Rui, D. Tao, Click prediction for web image reranking using multimodal sparse coding, Image Processing, IEEE Transactions on 23 (5) (2014) 2019–2032.

[27] A. K. Seth, A matlab toolbox for granger causal connectivity analysis, Journal of Neuroscience Methods 186 (2) (2010) 262–273.

[28] A. Prinzie, D. Van den Poel, Dynamic Bayesian Networks for Acquisition Pattern Analysis: A Financial-Services Cross-Sell Application New Frontiers in Applied Data Mining, Vol. 5433 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 123–133.

[29] A. C. Rencher, Methods of multivariate analysis, 2nd Edition, 2002.

[30] H.-T. Chen, C.-L. Chou, T.-S. Fu, S.-Y. Lee, B.-S. P. Lin, Recognizing tactic patterns in broadcast basketball video using player trajectory, Journal of Visual Communication and Image Representation 23 (6) (2012) 932 – 947.

[31] E. Castillo, J. M. Gutierrez, A. S. Hadi, Expert systems and probabilistic network models, 1996.

[32] N. Friedman, M. Linial, I. Nachman, D. Pe'er, Using bayesian networks to analyze expression data, Journal of Computational Biology 7 (3-4) (2000) 601–620.

[33] E. Castillo, U. Kjrulff, Sensitivity analysis in gaussian bayesian networks using a symbolic-numerical technique, Reliability Engineering and System Safety 79 (2) (2003) 139 – 148.

[34] J. Wagner, G. Stolovitzky, Stability and time-delay modeling of negative feedback loops, Proceedings of the IEEE 96 (8) (2008) 1398–1410.

[35] J. Wagner, L. Ma, J. J. Rice, W. Hu, A. J. Levine, G. A. Stolovitzky, p53-mdm2 loop controlled by a balance of its feedback strength and effective dampening using atm and delayed feedback, Systems Biology, IEE Proceedings 152 (3) (2005) 109–118.

[36] N. Friedman, K. Murphy, S. Russell, Learning the structure of dynamic probabilistic networks, Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98) (1998) 139–14.

[37] G. A. Fink, S. O. service), Markov models for pattern recognition from theory to applications (2008).

[38] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Computation 14 (8) (2002) 1771 – 1800.

[39] Y. Machigashira, Manifolds with pinched radial curvature, Proceedings of the American Mathematical Society 118 (3) (1993) pp. 979–985.

[40] B. Bocsi, L. Csato, J. Peters, Alignment-based transfer learning for robot models, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, 2013, pp. 1–7.

[41] D. M. Chickering, D. Heckerman, Efficient approximations for the marginal likelihood of bayesian networks with hidden variables, in: Machine Learning, 1997, pp. 29–181.

[42] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, Neural Networks, IEEE Transactions on 13 (2) (2002) 415–425.

[43] K. Y. H. Chow Jia Yi, Iti Chaturvedi, An analysis of herding behaviours in basketball as a function of skill level., in: ECSS, 2013.

Table 1: Results of regulations predicted by MVAR, SVM, DNN, and DTNN on synthetic time series datasets with up to third order delays. Mean and variance of TP, precision, recall and F-measure over five different test datasets have been reported for training time series of length (100/30) and Gaussian networks of size (10/20). The first column gives the number of time samples (T) used. The second column gives the number of nodes (N) in the target class network.

T	$\#N$	model	TP	Precision	Recall	F-measure
100	10	MVAR	15.67± 1.37	0.79± 0.07	0.78± 0.07	0.78± 0.07
		SVM	14.5± 3.39	0.73± 0.18	0.73± 0.18	0.73± 0.18
		DNN	13.83± 1.72	0.7± 0.08	0.70± 0.08	0.70± 0.08
		DTNN	**16± 1.26**	**0.82± 0.06**	**0.80± 0.06**	**0.80± 0.06**
	20	MVAR	14± 1.67	0.71± 0.08	0.70± 0.08	0.70± 0.09
		SVM	12.5± 2.35	0.65± 0.12	0.63± 0.12	0.62± 0.12
		DNN	12.17± 1.47	0.62± 0.08	0.61± 0.07	0.61± 0.08
		DTNN	**16± 1.55**	**0.82± 0.07**	**0.78± 0.11**	**0.79± 0.09**
30	10	MVAR	13.33± 0.82	0.67±0.04	0.67±0.04	0.66±0.04
		SVM	14.83± 2.14	0.75± 0.1	0.74± 0.11	0.74± 0.1
		DNN	12.5± 3.02	0.63± 0.15	0.63± 0.15	0.63± 0.15
		DTNN	**15.33± 1.21**	**0.77± 0.06**	**0.77± 0.06**	**0.77± 0.06**
	20	MVAR	10.5± 1.38	0.52± 0.07	0.52± 0.07	0.52± 0.07
		SVM	11.67± 1.97	0.62± 0.13	0.58± 0.1	0.57± 0.1
		DNN	11.5± 3.89	0.58± 0.2	0.58± 0.19	0.57± 0.2
		DTNN	**13.17± 1.94**	**0.66± 0.1**	**0.69± 0.1**	**0.67± 0.1**

Table 2: Computation time required to predict ML probabilities for synthetic Gaussian networks of size $N \in \{10/20\}$. The first column gives the number of nodes (N) in the target class network. The wall clock and time savings is reported for with and without use of transfer learning in seconds and minutes.

$\#N$	No Transfer	**Transfer**	Time Savings
10	48 mins	38 mins	25%
20	28 hrs	206 mins	635%

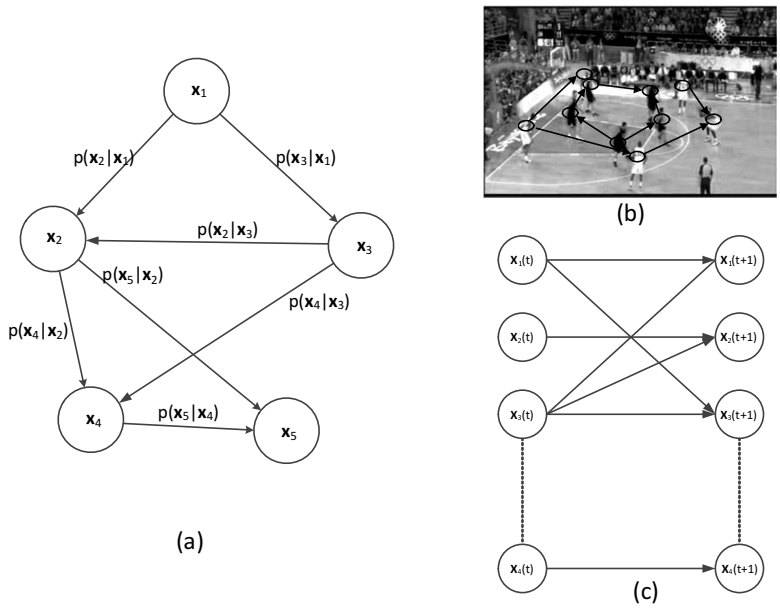


Figure 1: (a) Bayesian network for a multivariate system with five variables. Each node represents a Gaussian variable which can take continuous states from a normal distribution with mean μ and co-variance Σ. The connections represent causal dependencies in a single time instant. The observed state vector of variable i is denoted as $\mathbf{x}_i$ and the regulation or conditional probability of variable i given variable j is given by $p(\mathbf{x}_i|\mathbf{x}_j)$. (b) The Gaussian network describes a team of five Basketball players. The black nodes denote players of the defending team while the white nodes denote players of the offending team [30]. (c) The dynamic form of a Bayesian network assumes a first-order Markov assumption, that nodes are regulated by parents from a previous time-instant and learns the transition network between them.

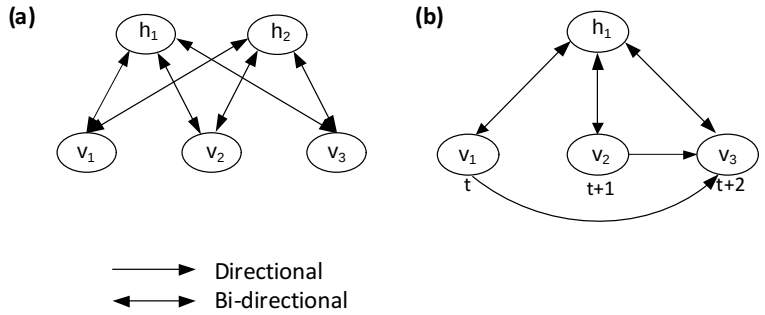


Figure 2: State-space diagrams for DNN. Edges can be directed or bi-directional. All bi-directional edges are learned using contrastive divergence. (a) Conventional RBM with two hidden neurons and three nodes in the visible layer, no edges are available between neurons in the same layer. (b) RBM for Gaussian networks with time delays. It has a single hidden neuron and three visible nodes. The visible nodes correspond to a Gaussian network with three variables. The directed edges are estimated by training the weights of the connections between visible nodes and hidden neuron from the training time series data.

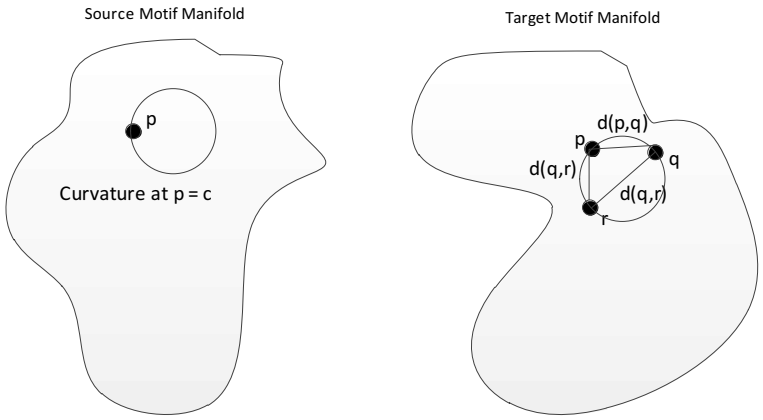
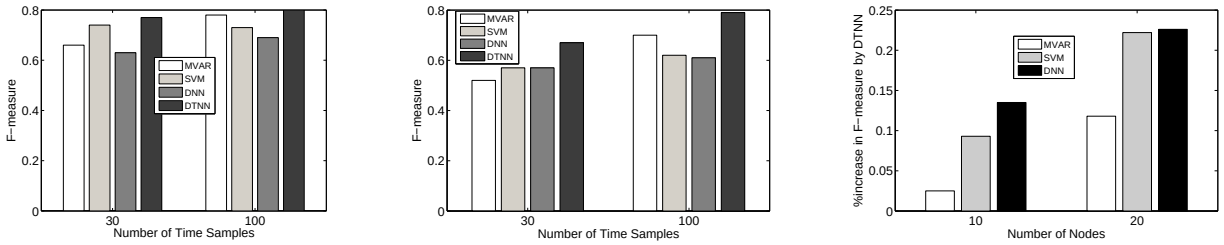


Figure 3: Illustration of the source motif and the target motif manifolds. We can determine if the manifolds are mappable using the inequality (10). The inequality constrains the curvature c at point p in the source manifold and the diameter of sphere passing through p and two other arbitrary points q and r in the target manifold. Details of the proof are available in [39].



(a) (b) (c)

Figure 4: Results on the predicted synthetic Gaussian networks. (a) Comparison of the F-measure for a DTNN with the baselines on the smaller Gaussian network of 10 nodes when 30 and 100 time samples are available is presented. (b) The F-measures of the various methods considered on the larger Gaussian network of 20 nodes are depicted. (c) The percentage of improvements in F-measure as attained by DTNN over MVAR, SVM and DNN on Gaussian networks of size 10 and 20 when 100 time samples are available, are depicted.

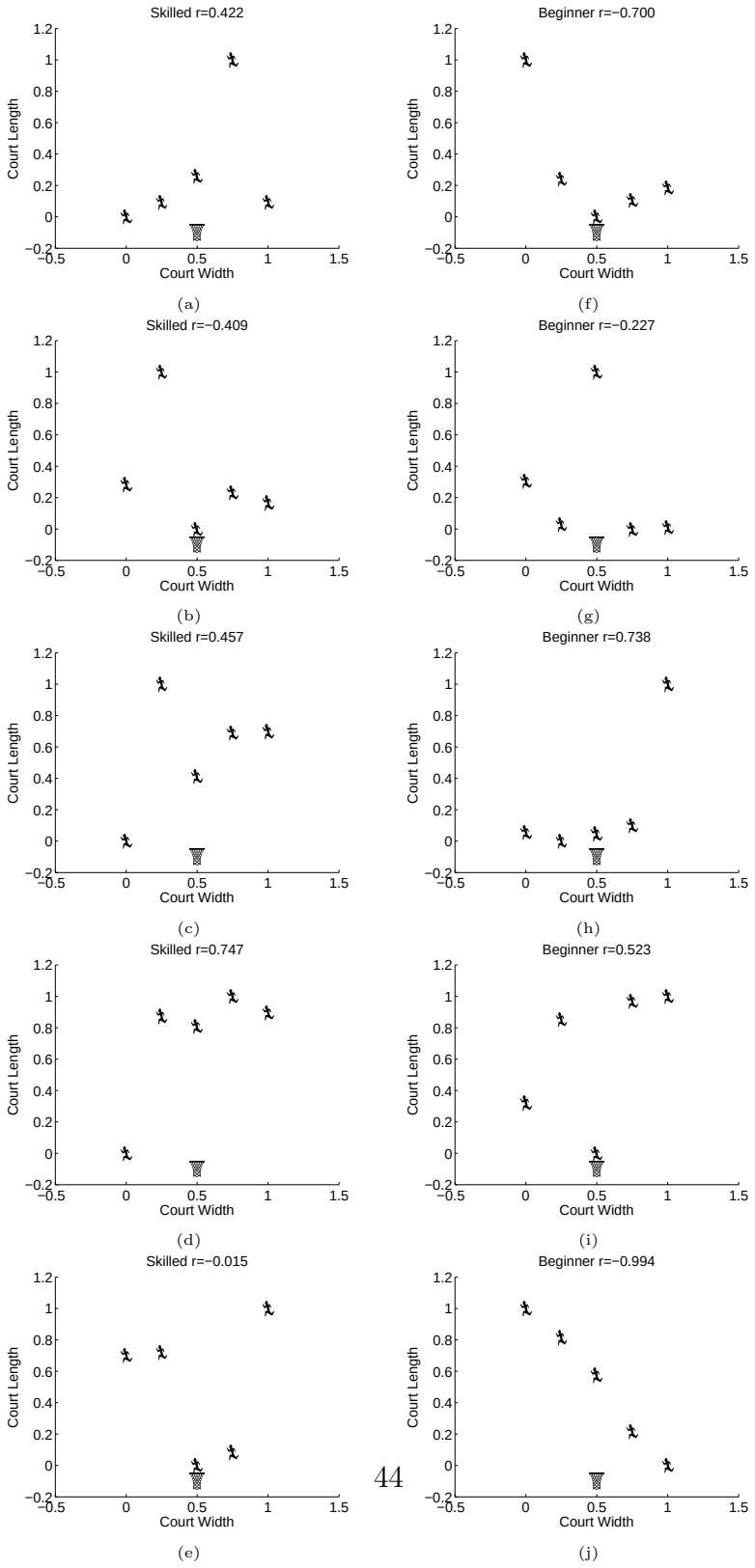


Figure 5: The predicted offensive behaviours and formations involving the five-member teams of Skilled and Beginner players in the game of Basketball are depicted.