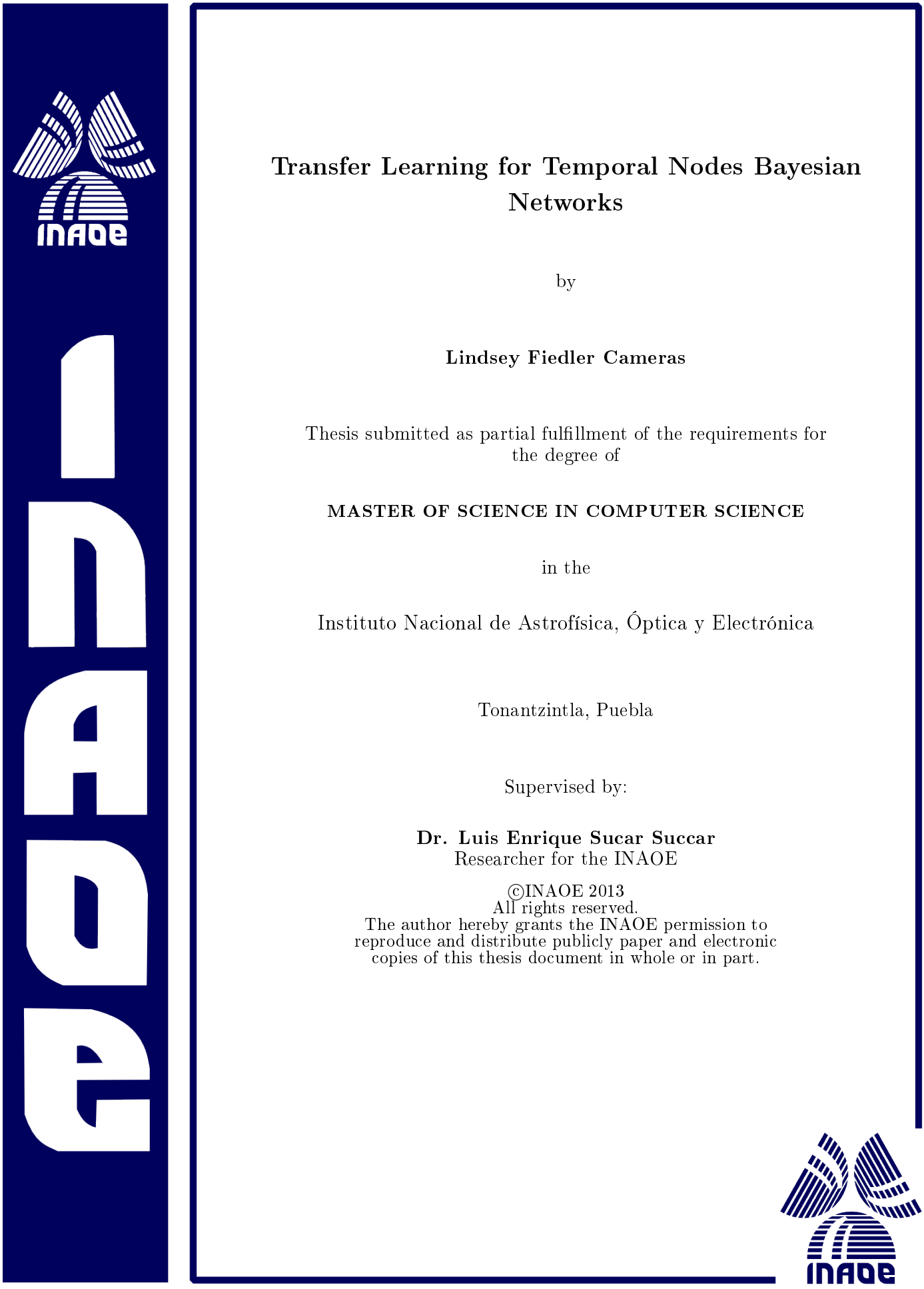


Transfer Learning for Temporal Nodes Bayesian Networks

by

Lindsey Fiedler Cameras

Thesis submitted as partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

in the

Instituto Nacional de Astrofísica, Óptica y Electrónica

Tonantzintla, Puebla

Supervised by:

Dr. Luis Enrique Sucar Succar
Researcher for the INAOE

Abstract

Probabilistic graphical models are a common choice for modeling domains that operate under uncertainty. Within this framework, Bayesian networks have become very popular due to their highly intuitive representation that makes their interpretation easy for someone unfamiliar with computational models. Temporal Nodes Bayesian Networks extend this concept by including a dynamic component to the model. Specifically, they model the relationships existing between events and when they occur. Recently, an algorithm for inducing a temporal nodes Bayesian network from a database was defined. This algorithm, denominated LIPS (Learning Interval, Parameters and Structure), implemented a novel strategy for learning the dynamic aspects of the model, while using traditional Bayesian network learning techniques for the structure and the parameters. Unfortunately, the results obtained by the LIPS algorithm depend heavily on the amount of data available for learning, and just as with many other traditional machine learning algorithms, learning from scarce data sets leads to unreliable models that produce poor results. Transfer learning is a paradigm that compensates for scarce data sets by borrowing information from other models that have already been learned. These auxiliary models differ from the target learning task, but hold some degree of similarity with it, such that the learning of certain aspects of the target model can be aided by this information transfer. Previously, work has been done for transfer learning in the area of probabilistic graphical models; however, the incorporation of transfer learning to dynamic models is relatively unexplored. In this thesis, a methodology for inducing a Temporal Nodes Bayesian Network using transfer learning is proposed. The algorithm defines a separate strategy for learning each component, where transfer learning is applied to induce the dynamic aspects of the model, as well as the structure and the parameters. Experimentation was carried out to evaluate the algorithm, and

the results were compared to those obtained without transfer and by applying naive transfer. Results show that the proposed algorithm obtains models that are significantly better than those obtained by learning only from scarce data sets (without transfer). In addition, they show that the algorithm is able to retrieve reliable models even when few records are available for the task of interest. The proposed algorithm was also applied to the real-world medical domain of the Human Immunodeficiency Virus (HIV) to learn mutational networks that develop as a response to antiretroviral therapy. These experiments were evaluated with the help of an expert in the field of HIV studies.

Resumen

Los modelos gráficos probabilísticos son una elección común para representar dominios que manejan incertidumbre. Dentro de los modelos pertenecientes a esta clase, las redes bayesianas se han vuelto muy populares gracias a que brindan una representación muy intuitiva que hace que su interpretación sea fácil para alguien sin experiencia con modelos computacionales. Las Redes Bayesianas de Nodos Temporales extienden el concepto de una red bayesiana al incluir un componente dinámico. Estas redes modelan las relaciones existentes entre eventos y cuándo ocurren. Recientemente se definió un algoritmo para inducir una red bayesiana de nodos temporales a partir de una base de datos. Este algoritmo, denominado LIPS por sus siglas en inglés (Learning Interval, Parameters and Structure), implementa una estrategia novedosa para aprender los aspectos dinámicos del modelo y utiliza estrategias tradicionales de aprendizaje de redes bayesianas para inducir la estructura y los parámetros. Desafortunadamente los resultados obtenidos por el algoritmo LIPS dependen en gran medida de la cantidad de datos disponibles para el aprendizaje. Al igual que con otros algoritmos tradicionales de aprendizaje de máquina, el aprender a partir de pocos datos genera modelos poco confiables con un desempeño pobre. El aprendizaje por transferencia es un paradigma que compensa la falta de datos, reutilizando información perteneciente a otros modelos que ya han sido aprendidos. Estos modelos auxiliares difieren de la tarea objetivo, pero mantienen un grado de similitud con ella, de tal manera que pueden ayudar a aprender ciertos aspectos del modelo objetivo. Mientras que se han hecho trabajos de aprendizaje por transferencia para modelos gráficos probabilísticos, el incorporar transferencia a un modelo dinámico es relativamente nuevo. En esta tesis se propone una metodología para inducir una Red Bayesiana de Nodos Temporales utilizando transferencia de conocimiento. El algoritmo define una estrategia para cada uno de los componentes del mod-

elo, donde se aplica transferencia de conocimiento para inducir tanto los aspectos dinámicos como la estructura y los parámetros. Se realizaron varios experimentos para evaluar el algoritmo, y se compararon los resultados con los obtenidos al aprender sin transferencia y con transferencia simple o *naive*. Los resultados muestran que el algoritmo propuesto obtiene modelos significativamente mejores que los que se obtienen al aprender solamente de los pocos datos, es decir sin transferencia de conocimiento. Además, muestran que el algoritmo es capaz de aprender modelos confiables aun cuando hay pocos datos disponibles para la tarea objetivo. El algoritmo propuesto también se aplicó al dominio médico del Virus de la Inmunodeficiencia Humana (VIH) para aprender vías mutacionales que se generan como respuesta al tratamiento con antiretrovirales. Estos experimentos fueron evaluados con la ayuda de un experto en el estudio de VIH.

Acknowledgments

I would like to thank my supervisor, Dr. L. Enrique Sucar Succar, for all the time and guidance he provided to complete this work. I would also like to thank Dr. Eduardo Morales Manzanares; although he was not my official supervisor, his guidance proved to be equally invaluable.

To my thesis committee, Dr. Hugo Jair Escalante Balderas, Dr. Enrique Muñoz de Cote Flores Luna and Dr. Leopoldo Altamirano Robles, I would like to extend my gratitude for their astute observations and the time taken to evaluate my work.

For his collaboration in my experimentation with the HIV medical domain, I would like thank Dr. Santiago Ávila. His insights proved to be vital for the completion of these experiments.

I am especially grateful to the Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), and in particular to the Computational Sciences Coordination and their wonderful personnel of professors and researchers. With the teachings they provided during the first year of my studies, they have given me the tools to accomplish new and exciting endeavors.

For the financial support they provided me with during my studies, I would also like to extend my gratitude to the Consejo Nacional de Ciencia y Tecnología (CONACyT).

To the friends I have made during my time at the INAOE, I would like to thank you all for your company and for providing me with the sometimes necessary distractions. It has truly been my pleasure to get to know all of you.

Finally, a special thank you goes to my family for always giving me their love and support. It is because of them that I can achieve my dreams.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Within many domains there is an emerging need for the creation of models that are representative of a particular problem. What is desired is to be able to leverage the knowledge available about a certain task, so that the most meaningful patterns may be extracted to create a model able to make accurate predictions. The central objective of machine learning focuses precisely on creating systems capable of learning from experiences without the need to be explicitly programmed. The experiences from which models are learned are typically data relating to examples of past behaviors, and the goal is to be able to generalize from them so that when a new and previously unseen example occurs, the model is able to perform accurately.

Unfortunately, the data available for a problem is not always correct or complete, resulting in uncertainty. In these situations, probabilistic graphical models [KF09] have become a popular choice for domain modeling thanks to their solid foundation in probability theory, and their highly intuitive graphical representation. One such model are Bayesian Networks, which model the relationships that occur between variables of a problem domain using nodes and arcs. In a Bayesian network, random variables are represented by nodes, and the relationships between them with arcs. This simple and concise representation makes them a common choice for modeling domains with uncertainty, and many algorithms for learning

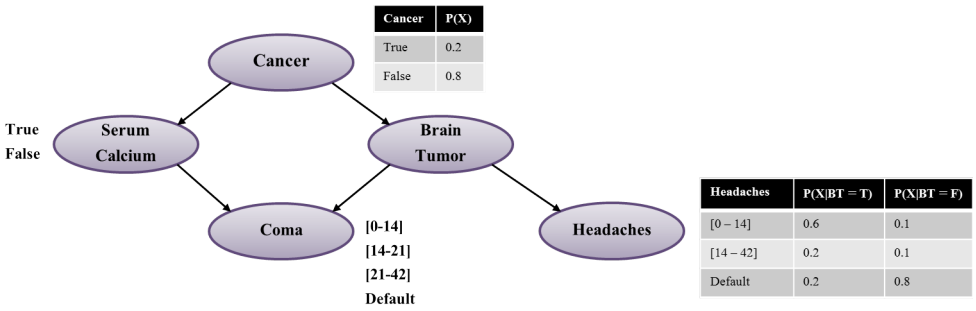


Figure 1.1: An example of a Temporal Nodes Bayesian Network. Each node represents an event, and the arcs (arrows) between them temporal relationships. A probability distribution is associated to every node. For simplicity only the probability tables for "Cancer" and "Brain Tumor" are shown.

them from data have already been proposed [CH92, SGS00].

In addition to uncertainty, some problem domains are dynamic in nature, meaning that they contain information that changes over time. For these types of domains it is not only interesting to model the relationships that occur between variables, but also *when* they occur. Dynamic probabilistic graphical models offer a representation to domains of dynamic behavior while also keeping present their uncertain nature. One type of dynamic probabilistic graphical model is the Temporal Nodes Bayesian Network (TNBN). TNBNs offer a compact visual representation for the problem domain and are an effective option to represent dynamic processes. An example of a TNBN modeling the outcomes of being diagnosed with metastatic cancer is presented in Figure 1.1. In this example, the root node "Cancer" has a 20% probability of occurring, and it triggers two events, an increase in serum calcium and a brain tumor. These events can then lead to a coma or to the development of headaches; however, these symptoms do not necessarily occur immediately but rather within a time range. Both "Coma" and "Headaches" have a set of temporal intervals associated to them representing when the event occurs, or a "Default" value if it does not occur.

Unfortunately, some difficulties can occur when learning models. Aside from uncertainty being present, the data from which a model is learned can also be scarce. In these cir-

cumstances extracting meaningful patterns becomes nearly impossible, and the results of learning under these conditions are unreliable models that are incapable of making accurate predictions.

Transfer learning offers a solution to the problem of having scarce data by reusing knowledge learned previously for other tasks. The basic idea behind transfer learning is to compensate for the lack of information by extracting useful knowledge from other different, but closely related, domains; and then applying this knowledge when learning the task of interest. Interest in algorithms that apply transfer learning has increased because they eliminate the need to be constantly collecting and labeling new data. For example, in domains where information quickly becomes outdated, transfer learning can be applied to learn updated models.

Recent work done with transfer learning suggests that it is indeed a viable approach for learning models when data is scarce. However, this strategy has yet to be applied to the learning of a Temporal Nodes Bayesian Network. Behind the success that other models have seen by using transfer learning, the central claim of this work was formed:

"Transfer learning will improve the learned Temporal Nodes Bayesian Network models when the data available for training is scarce, if closely related auxiliary domains learned with sufficient data are used as sources for transfer."

1.1 Research Challenges

Previous efforts have used transfer learning for inducing probabilistic graphical models [NMC07, LSM10, ODW01]. Particularly, the Bayesian network model has been a popular choice for learning with transfer mainly because of the wide variety of existent algorithms for learning and performing inference. However, since Bayesian networks do not consider temporal information, the transfer learning strategies developed to learn them cannot be adequately applied to dynamic models like the Temporal Nodes Bayesian Network model. In fact, learning a TNBN poses a more challenging learning problem than Bayesian networks, because in

addition to the structure and parameters, the temporal intervals must also be induced.

Furthermore, transfer learning presents challenges of its own. In addition to questions of *how to transfer*, there are also questions of *what to transfer* and *when to transfer*. If knowledge is transferred indiscriminately, it is possible for transfer learning to impact the learning negatively, resulting in decreased performance. To avoid a negative impact, a transfer learning method should identify those domains that are closest to its target task, and correctly leverage the pertinent information.

This thesis proposes a methodology for fully learning a TNBN with transfer learning. The proposed algorithm incorporates novel approaches to transfer knowledge from several auxiliary source TNBNs to learn the structure, parameters and intervals of a target TNBN. The transfer learning algorithms designed for each component limit the possibility of negative transfer by leveraging the similarities between domains. Specifically, domains with a weak relationship to the target domain are limited in the amount of knowledge they transfer, while domains with stronger relationships transfer more generously.

1.2 Objectives

The objective of this thesis is to develop an algorithm that uses inductive transfer to learn a Temporal Nodes Bayesian Network by learning from several related auxiliary domains and the scarce data available for the target domain. It is expected that the resulting models show a predictive precision superior to what would be obtained if the models were learned solely from the scarce target data, that is, without applying transfer learning.

1.2.1 Specific Objectives

To accomplish the objective stated above the following specific objectives are identified:

- Measure the strength of the relationship between the target domain and an auxiliary source domain.

- Develop a transfer learning algorithm for learning the temporal intervals associated to the nodes of a TNBN.
- Modify the PC-TL algorithm [LSM10] to learn the structure of a TNBN.
- Develop a transfer learning algorithm for learning the parameters of a TNBN.
- Evaluate the learned models using synthetic data and real world data related to the medical treatment of the Human Immunodeficiency Virus (HIV).

1.3 Main Contributions

The main contribution of this thesis is an algorithm that employs transfer learning to compensate for the lack of training data when learning a Temporal Nodes Bayesian Network. To fully learn a TNBN, a strategy for inducing each of its components must be defined. Therefore, the contributions of this thesis can be broken down as follows:

- The definition of an algorithm that uses transfer learning for inducing the temporal intervals associated to the nodes of a TNBN.
- The adaptation of the PC-TL algorithm to use a node ordering when inducing the structure of a graphical model.
- The definition of an algorithm that applies transfer learning to learn the parameters for the nodes of a TNBN.

To evaluate the proposed algorithm, experiments with two data sets were performed. The first experiments used synthetic data for learning the models, while the second experiments used real world data. The models learned with the proposed algorithm were compared to models learned without transfer learning, and to models learned by performing naive transfer. Mainly three elements were evaluated and compared: the predictive precision of the models, the accuracy in estimating the temporal events, and the accuracy of the induced

structure compared to that of the expected structure. This last element was only evaluated in situations where a gold standard exists for comparison.

Experimental results show that Temporal Nodes Bayesian Network models learned with transfer learning are significantly better in comparison to those learned purely from small data sets, that is, without transfer. When comparing with models learned with naive transfer, results show that the proposed algorithm is capable of avoiding negative transfer by leveraging the existent similarities between domains, and avoiding those elements which are more disparate. Overall, the experiments show that the proposed algorithm is able to retrieve a reliable model even when few records are available for the target domain. This supports the claim of this thesis that transfer learning is a viable strategy for learning Temporal Nodes Bayesian Networks when training data is scarce.

Lastly, another contribution is the application of transfer learning to dynamic probabilistic graphical models, an area that has been underexplored thus far.

1.4 Thesis Structure

This thesis is organized as follows. In Chapter 2 the Bayesian network model is presented. Understanding this model is the basis for understanding the Temporal Nodes Bayesian Network model which is the focus of this thesis, and is formally presented in Chapter 3. In Chapter 4 transfer learning is discussed, along with previous work done for probabilistic graphical models. Chapter 5 gives a detailed explanation of the proposed methodology for learning a Temporal Nodes Bayesian Network with transfer learning, and in Chapters 6 and 7 an experimental evaluation of this methodology is presented were models are learned from synthetic data relating to the event of an automobile accident, and real world data about the medical treatment of the Human Immunodeficiency Virus. Finally, in Chapter 8 conclusions about the work done in this thesis are given, along with some suggestions for future lines of investigation.

Chapter 2

Bayesian Networks

In this chapter the Bayesian network model is presented, as this model is the foundation for understanding temporal nodes Bayesian networks. A formal definition for the model is provided, along with an example of a Bayesian network. Subsequently, the procedure for learning the structure and the parameters of a Bayesian network is discussed. Finally, two extensions of the Bayesian network model are presented.

2.1 Formal Definition

A Bayesian network [Pea88] is a probabilistic graphical model which uses a directed acyclic graph, or DAG to represent conditional independencies over a set of random variables. A solid foundation in probability theory, coupled with the ability to perform bidirectional inferences has made them a popular choice to model domains where uncertainty is present. In addition, Bayesian networks provide a very intuitive visual representation of the domain they are modeling.

In a Bayesian network, the nodes of the DAG represent variables of interest, and the arcs, the statistical dependencies between them. An arrow between two nodes implies a relationship, where the node spawning the arrow is a parent to the node to which it points. The value taken by a variable is therefore independent of its non-descendants given its parents.

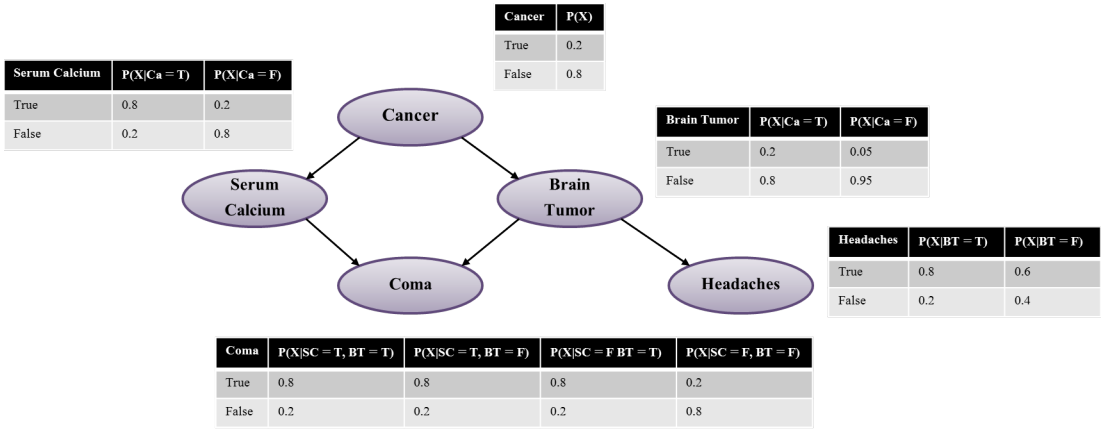


Figure 2.1: A Bayesian network representing the outcomes of being diagnosed with metastatic cancer. Variables are represented with nodes, while dependency relations between them are represented with arcs. The node from which an arc spawns is a catalyst to the effect presented in the node to which that arc points. Probability tables, quantifying the dependency relations, are provided for all variables.

This property, known as the Markov property, allows the model to be described succinctly with conditional probability distributions for each node.

Formally, a Bayesian network is defined by two components:

- A directed acyclic graph $G = (V, E)$ where V is the set of nodes present in the network, and E is the set of arcs connecting those nodes such that no cycles are formed. An element of E is defined as an ordered pair of members of V.
- A collection of parameters Θ, which quantify the network. These parameters are the conditional probability distributions $P(X|Pa(X))$, where X is a random variable belonging to the network, and $Pa(X)$ is the set of its parents.

Figure 2.1 displays an example of a Bayesian network which models the outcomes of being diagnosed with metastatic cancer (an advanced stage of cancer) as presented in [Fas10]. Cancer is a possible cause of a person developing a brain tumor, and can also provoke an increase in serum calcium. Both a brain tumor and an increase in serum calcium can lead to a person falling into a coma. In addition, a brain tumor may also provoke severe headaches.

The example network of Figure 2.1 consists of five random variables modeled by the nodes "Cancer" (Ca), "Serum Calcium" (SC), "Brain Tumor" (BT), "Coma" (Co) and "Headache" (H). A joint probability distribution that describes the network is expressed as follows:

$$P(Ca, SC, BT, Co, H) \tag{2.1}$$

Using the Markov property, the joint probability distribution can be rewritten as the product of the probabilities of each variable given its parents.

$$P(Ca, SC, BT, Co, H) = P(Ca)P(SC|Ca)P(BT|Ca)P(Co|SC, BT)P(H|BT) \tag{2.2}$$

2.2 Learning Bayesian Networks

Learning a Bayesian network consists of defining the structure and the parameters that quantify the network. One approach to building a Bayesian model involves eliciting knowledge from an expert on the domain. This task is neither easy nor quick, and only becomes more difficult as the number of random variables in the network increases. A more popular approach is to learn the model from available data about the domain. This strategy uses machine learning to induce the structure and the parameters of the model. In addition, the available data can be combined with expert knowledge to produce more reliable models.

A number of algorithms have been proposed for inducing Bayesian networks from data [CH92, SGS00, HGC95]. These can be divided into algorithms for structural learning and algorithms for parametric learning.

2.2.1 Structural Learning

The objective of structural learning is to find the DAG that best represents the conditional dependencies present in the data. One alternative for learning the structure is to combine knowledge from an expert with the available data. In this approach the expert provides an

initial structure which is subsequently validated and improved with statistical tests performed on the data.

Another alternative is to learn the structure purely from the available data. This approach assumes that the data is a reliable representation of the *true* probability distribution for the model, and therefore a learning algorithm can obtain the optimal DAG. In general, the process of learning the structure can be seen as a search among the space of possible DAGs, however as the number of variables in the model increases, it becomes impossible to do an exhaustive search over all possible structures. It has been proven that finding the best structure is NP-complete [Chi96], for this reason structural learning algorithms use strategies to limit the search space, and therefore may only return an approximation to the optimal DAG.

Structural learning algorithms can be divided into two categories:

1. Score and search based methods.
2. Constraint based methods.

Score and Search Methods

Score and search methods explore the space of possible DAGs by assigning a score to a candidate DAG and then attempting to maximize this value through a series of alterations to the structure, i.e., removing, adding, or inverting arcs. Algorithms of this category use a heuristic to guide the search, where the score of a structure is an evaluation of how well the current DAG describes the data. Many evaluation metrics exist, however the most commonly used are based on the Bayesian score and the minimum description length principal.

The K2 Algorithm

The K2 algorithm [CH92] is an example of a score and search method that uses a Bayesian score. This algorithm applies a *greedy* approach to obtain a structure consistent with the data by using information about the causal ordering. Since parent nodes always precede child nodes in the causal ordering, knowing the correct order of the nodes reduces the problem to selecting the best set of parents for each node, a problem which can be solved in polynomial time.

The K2 algorithm assumes that a total ordering on the nodes is available. Initially, each node has no parents. The algorithm iterates through the nodes using the provided order, and incrementally adds as parents, the nodes that most increase the score of the structure. The score of a structure is the probability of the data given the selected parents for that particular node, and is calculated using the following function:

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \tag{2.3}$$

where:

- π_i is the set of parents belonging to node x_i.
- q_i is the number of possible instantiations the parents of x_i can take.
- r_i is the number of values x_i can take.
- N_{ij} is the number of cases present in the data where the parents of x_i are instantiated with their j^{th} value.
- α_{ijk} is the number of cases present in the data where x_i is instantiated with its k^{th} value, and the parents of x_i are instantiated with their j^{th} value.

Algorithm 2.1 The K2 Algorithm

Require: Set of nodes n, Ordering on the nodes o, Upper bound u on the number of parents a node may have
Ensure: Directed Acyclic Graph G
  for $i = 1$ **to** n **do**
    $\pi_i = \emptyset$
    $P_{old} = f(i, \pi_i)$ (see Equation 2.3)
    $Proceed = true$
    while $Proceed$ and $|\pi_i| < u$ **do**
      let z be the node in $Pred(x_i) - \pi_i$ that maximizes $f(i, \pi_i \cup \{z\})$
      $P_{new} = f(i, \pi_i \cup \{z\})$
      if $P_{new} > P_{old}$ **then**
        $P_{old} = P_{new}$
        $\pi_i = \pi_i \cup \{z\}$
      else
        $Proceed = false$
      end if
    end while
    Set directed arcs in G from all nodes in π_i to x_i
  end for
  return G

The procedure stops when adding a node no longer increases the score of the structure. An upper bound on the number of parents each node may have can be provided as an additional constraint. Algorithm 2.1 provides the basic procedure for the K2 algorithm[1].

The K2 algorithm makes a strong assumption in that the node ordering provided is the correct causal ordering. If the ordering is unknown, one can be randomly selected, however this can lead the algorithm to produce a less reliable structure. A more appropriate alternative is to search over all possible node orderings [TK05]. This search space is much smaller than the space of possible DAGs, and thus more efficient.

[1] $Pred(x)$ is the set of nodes that can be predecessors to x given the ordering.

Constraint Based Methods

Constraint based methods work on the understanding that a DAG models the conditional dependencies (and independencies) present in the data. Methods of this type, constrain the search space of possible DAGs with conditional independence statements, and subsequently attempt to retrieve the DAG that best satisfies these constraints. To determine conditional independence between two variables, statistical tests are performed on the data; where based on the results of these tests, arcs between variables are either removed or added. Some examples of algorithms that apply a constraint based approach are the Chow-Liu algorithm for learning trees [CL68], and the PC algorithm [SGS00].

The PC Algorithm

The PC algorithm is a constraint based method that recovers a DAG by measuring the statistical dependence between variable pairs. Several important assumptions are made by this algorithm:

1. Causal sufficiency: There are no unobserved or hidden variables in the domain that are parents to other observed variables.

2. The Markov property: A variable is independent of all of its non-descendants given its parents.

3. Faithfulness: The independence relations represented in the retrieved DAG are all and only the independence relations present in the probability distribution.

Additionally, the PC algorithm makes the assumption that the available data is sufficient for performing accurate statistical tests, and that a procedure exists to uncover independencies. One alternative for this procedure is the conditional cross entropy measure used in conjunction with a threshold value representing statistical confidence.

Algorithm 2.2 The PC Algorithm

Require: Set of variables X, Independence test I
Ensure: Directed Acyclic Graph G
  Initialize a completely connected undirected graph G'
  $i = 0$
  repeat
    for $X \in \mathbf{X}$ **do**
      for $Y \in ADJ(X) - \{Y\}$, $|S| = i$ **do**
        if $I(X, Y|S)$ **then**
          Remove the arc $X - Y$ from G'
        end if
      end for
    end for
    $i = i + 1$
  until $|ADJ(X)| \leq i, \forall X$
  $G =$Orient arcs in G'
  return G

Algorithm 2.2 provides an outline for the PC algorithm[2]. The procedure done by PC can be divided into two phases. In the first phase, the algorithm begins with a fully connected undirected graph. It then iterates between all adjacent variable pairs X, Y to determine if they are independent given some subset S of other variables, i.e., $I(X, Y|S)$. If two variables are found to be independent, the arc between them is removed. The result of this first phase is the skeleton of the structure, that is, the underlying undirected graph.

The second phase of the algorithm orients the arcs to obtain the final DAG. In this step, the directions of the arcs are set based on conditional independence tests between variable triplets. It begins by looking for all substructures of the form $X - Z - Y$, where no arc exists between X and Y, and Z does not belong to the subset S that makes X and Y independent. A V-structure is then created by orienting the arcs as $X \rightarrow Z \leftarrow Y$. After all V-structures are found, the direction of some of the remaining arcs can be inferred based on the results of the independence tests, while taking care to avoid inserting cycles.

[2] $ADJ(X)$ is the set of nodes adjacent to X in the graph.

2.2.2 Parametric Learning

Parametric learning is the process of finding the probabilities associated with the obtained structure. For root nodes, the process consists of finding the *a priori* probabilities, while for child nodes it consists of finding the conditional probability distributions given the node's parents. When all the variables are observed, a maximum likelihood approach can be applied for calculating the probabilities. If hidden variables exist or there are missing values in the data, a different strategy must be taken. For data containing missing values, a simple approach is to assign the most probable value for that variable given the values of the other variables.

For hidden variables, a common approach is the Expectation-Maximization (EM) algorithm [Moo96]. The process consists of two steps: 1) the *E step* estimates probabilities for the hidden variables based on the current parameters; subsequently 2) the *M step* maximizes the parameters using the probabilities found in the E step. The procedure iterates between the two steps until a convergence criteria is met.

2.3 Bayesian Networks with Continuous Variables

Though traditionally Bayesian networks model discrete data, it is possible to include continuous variables. However, the presence of continuous variables adds significant difficulties to the model, as recognized by Pearl in [Pea88].

One such model that considers discrete and continuous variables is the *Conditional Linear Gaussian Bayesian Network* [KM08]. In this model, each continuous variable is modeled by a Gaussian density function, and discrete variables are modeled by probability tables. While this model includes a mixture of both discrete and continuous variables, it does impose the restriction that a discrete variable cannot have as a parent a continuous variable.

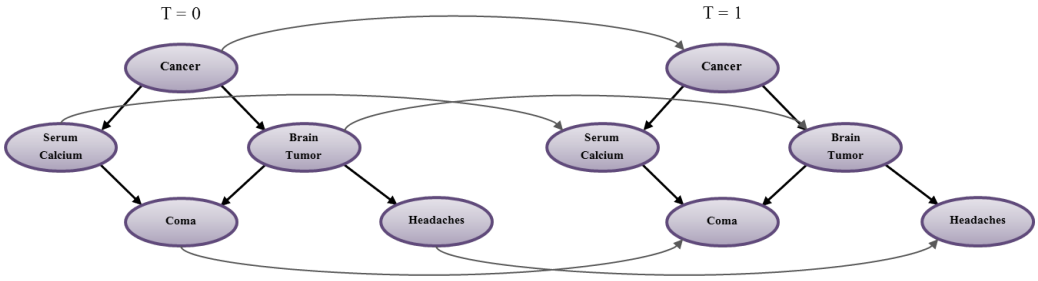


Figure 2.2: A Dynamic Bayesian Network modeling the outcomes of being diagnosed with metastatic cancer. The DBN models two time points, where each has a copy of the Bayesian network, or static network, and the copies are linked together with the transition network.

2.4 Dynamic Extensions

Dynamic extensions of Bayesian models exist to model processes with variables that undergo state changes through time. These models not only consider the relationships between variables, but also *when* a variable changes state. One such model is the Dynamic Bayesian Network, or DBN [Mur02].

DBNs model processes that occur over a range of time, and whose variables suffer many state changes. Like Bayesian networks, DBNs are probabilistic graphical models that can be used to make predictions on the future states of variables. A DBN is specified by two components: 1) a static network, and 2) a transition network. The static network is a Bayesian network which represents a process at a single point in time. To model the dynamic behavior of a process, the temporal range in which it occurs is discretized, and for each time point a copy of the static network is generated. The transition network links the copies together in order to capture the temporal relations between variables. When modeling dynamic information, DBNs obey the assumption that future states are conditionally independent from past states given the present state (Markov property); additionally they assume that the conditional probabilities which describe the temporal relations between random variables of adjacent time points do not change (stationary process). An example of a DBN is provided in Figure 2.2.

Other dynamic extensions of Bayesian models are event networks. In event networks, the value of a variable represents the time in which an event occurred. Unlike DBNs, the concern of these models is not the state to which the variable changes, but the point in time when the change occurs. In addition, their focus is on modeling irreversible events. This means that each variable can suffer at most one state change for the entire temporal range of interest, whereas DBNs can model several changes in state. An example of this type of model is the Temporal Nodes Bayesian Network model, or TNBN [AFS99]. Previously, TNBNs have been used to model mutational pathways in the Human Immunodeficiency Virus [HLRFÁR+13], and for predicting failures in a power plant [AFSV98, HLSG+11].

While DBNs focus on the state a variable is in at a given time, event networks are centered on when the state change occurs. In the next chapter the temporal nodes Bayesian network model is presented and described in more detail. This model is the focus of the present thesis.

2.5 Chapter Summary

In this chapter, the Bayesian network model was presented, and the necessary steps for learning a Bayesian network were reviewed. In particular, to learn the structure, score and search methods and constraint based methods were discussed, analyzing for each case, an algorithm of the given type. A brief overview of some dynamic extensions to Bayesian networks was given, providing an introduction to event networks, and thus laying the foundation for the next chapter. In the next chapter, the temporal nodes Bayesian network model is discussed. This model belongs to the event network category.

Chapter 3

Temporal Nodes Bayesian Networks

In this chapter, the Temporal Nodes Bayesian Network model is presented. This chapter begins by presenting a formal definition of the TNBN model, along with an example of this kind of network. Subsequently, the procedure for learning a TNBN is discussed, giving further emphasis on the LIPS algorithm for learning TNBNs.

3.1 Formal Definition

Temporal Nodes Bayesian Networks are event networks that present an alternative to dynamic Bayesian networks for modeling dynamic processes. They are an appropriate choice for representing processes where the variables in the temporal range of interest experience few state changes. For example, when a system failure occurs in a power plant, other subsystems (e.g., the feedwater pump, water steam generator, etc.) will begin to deviate from normal operation within minutes or hours. The change from normal operation to failure for each subsystem occurs only once, and therefore, fault diagnosis in a power plant can be successfully modeled with a TNBN [AFSV98, HLSG+11].

A TNBN is a probabilistic graphical model in which nodes of the graph represent events, and arcs between nodes represent temporal probabilistic relations. Two types of events are modeled in a TNBN: instantaneous and temporal. An instantaneous event is an event in

which no time delay is seen before its occurrence. In other words, once a parent event takes place the manifestation of the child event is immediate. A node that models an instantaneous event is defined to be an *instantaneous* node.

Nodes where the occurrence of the event is not immediate are modeled by temporal nodes (TNs). A TN models the possible time delays between the occurrence of the cause and the observation of the effect. A TN consists of a set of time intervals in which an event may take place, and a default value that indicates that the event did not occur. Since the occurrence of a temporal event requires a parent event to take place, all TNs are child nodes, and therefore all root nodes of a TNBN must be instantaneous. Formally, a temporal node is defined as follows:

Definition 3.1. *A temporal node is a random variable defined by a set of states each characterized by an ordered pair* (λ, τ) *where* λ *is the value taken by the random variable and* τ *is the temporal interval* $[t_i - t_f]$ *in which the state change occurred. A default state of no change that corresponds to the event "not occurring" is also associated to every temporal node. There is at most one state change for each temporal node in the temporal range of interest.*

Formally, a TNBN is defined as follows.

Definition 3.2. *Let* V *be a set of instantaneous and temporal nodes, and* E *a set of arcs between those nodes. A TNBN is a pair* $B = (G, \Theta)$ *where* G *is a directed acyclic graph (DAG),* $G = (V, E)$ *and* Θ *is a set of conditional probability distributions that quantify the network.*

Figure 3.1 extends the example provided in Figure 2.1 to model two temporal events. As in the original network, a diagnosis of metastatic cancer can lead to increased serum calcium and a brain tumor; however, the consequences of presenting these two symptoms may not be immediately apparent. Instead, a person may enter a comatose state or present headaches within some weeks of being diagnosed.

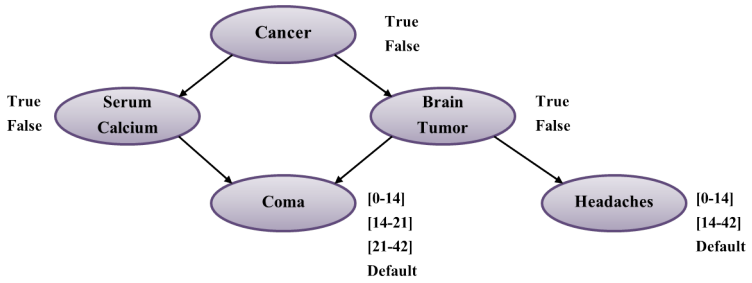


Figure 3.1: A temporal nodes Bayesian network modeling the outcomes of being diagnosed with metastatic cancer. "Cancer", "Serum Calcium" and "Brain Tumor" are instantaneous nodes, and "Coma" and "Headaches" are temporal nodes each with a set of intervals and a default value.

This example presents 3 instantaneous nodes (Cancer, Serum calcium and Brain tumor), and 2 temporal nodes (Coma and Headache) each with a set of temporal intervals associated. If a person presents an increase in serum calcium or a brain tumor, he or she may fall into a coma within 0 to 14 days, 14 to 21 days or 21 to 42 days, or not at all (Default). Additionally, if a person presents a brain tumor, he or she may begin to present headaches within 0 to 14 days, 14 to 42 days, or not at all (Default).

3.2 Learning a TNBN

TNBNs present a more challenging learning problem than Bayesian networks, because in addition to learning the structure and parameters, a procedure for learning the temporal intervals must also be defined. A simple approach for learning the temporal intervals associated to a TN would be to discretize the entire temporal range into k intervals of equal length. The problem then becomes finding a good value for k, as a small value could lead to poor predictive accuracy, and a large value to an overly complex network. This approach is characterized by generating intervals of equal granularity; however, this uniform discretization may not be the optimum choice for the task being modeled, and in these situations it would be desirable to be able to learn temporal intervals of variable granularity.

3.2.1 Friedman's Algorithm

Friedman and Goldszmidt proposed an algorithm for discretizing continuous variables and learning the structure of a Bayesian network [FG96]. While this algorithm was intended for learning static Bayesian networks, it could very well be applied for inducing the temporal intervals of a TNBN.

The basis of the algorithm is the scoring metric, which is based on the Minimum Description Length principle, or MDL. For their algorithm, Friedman and Goldszmidt propose an augmented version of MDL to take into account the discretization of the continuous variables. This new scoring metric will therefore be an evaluation of the complexity of the discretized variables and the learned network, and how well they model the data. Since MDL involves a tradeoff between model complexity and fitness, by including the complexity of the discretized continuous variables in the metric, the algorithm can find an adequate number of intervals to efficiently model the data.

3.2.2 The LIPS Algorithm

Hernandez-Leal et al. presented a procedure to learn a TNBN from data in [HLGMES13]. Their procedure denominated "Learning Intervals, Parameters and Structure" or LIPS, allows for the definition of multiple time intervals with different granularity. The LIPS algorithm defines a procedure for learning the structure, parameters and intervals of a TNBN. While common techniques for Bayesian networks are applied to learn the structure and parameters of the model, a novel procedure for inducing the temporal intervals belonging to each TN is defined, where a clustering algorithm is used to obtain a set of intervals for each TN. Specifically, the LIPS algorithm applies clustering to divide the temporal data for each TN into groups, and then induces the intervals from each group's parameters (i.e., its mean and standard deviation).

A brief description of two popular clustering algorithms, k-means and the Gaussian mixture model (GMM), is now provided.

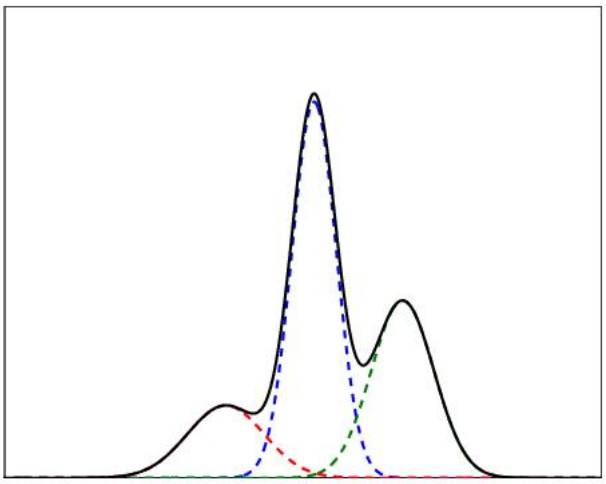


Figure 3.2: A Gaussian mixture model formed by three separate Gaussian distributions. This image is reproduced from [Cho02]. Individual Gaussian distributions are drawn in color, while the mixture of them is displayed in black.

The k-means Algorithm

One of the most widely known and simplest clustering algorithms is the k-means algorithm [HW79]. This algorithm receives as input a value k that represents the number of clusters to be obtained. The procedure begins by randomly initializing k centroids, one for each cluster. It then iterates between two steps: 1) assigning the remaining data points to the closest centroid, and 2) recalculating the centroids of each cluster based on the previous assignations. This process repeats until a convergence to a local optimum is reached.

The Gaussian Mixture Model

Like the k-means algorithm, the Gaussian mixture model is a clustering algorithm [Pre07]. The GMM algorithm has its foundation in statistical distributions, particularly the Gaussian distribution, also called the Normal distribution. GMM assumes that the data to be grouped is generated by a mixture of several Gaussian distributions. Figure 3.2 shows an example of

Algorithm 3.1 The LIPS Algorithm

Require: A data set D with temporal data
Ensure: A TNBN T
  Discretize the temporal data present in D
  while Convergence is not reached in structure **do**
    Learn the structure
    Refine the intervals
  end while

a GMM formed from three Gaussian distributions.

Formally, a GMM is a parametric probability density function represented as a weighted sum of M component Gaussian densities:

$$p(x) = \sum_{i=1}^{M} \pi_i g\left(x|\mu_i, \sigma_i^2\right) \tag{3.1}$$

where each $g\left(x|\mu_i, \sigma_i^2\right)$ is a Gaussian with a mean μ_i and a variance σ_i^2, and π_i is a mixing coefficient such that $0 \leq \pi_i \leq 1$ and $\sum_{j=1}^{m} \pi_j = 1$. The GMM is therefore determined by the parameters π, μ, σ^2.

Unlike in the case where the data come from only one distribution, in a GMM the distribution to which a data point contributes is unknown. Consequently, the formulas used for calculating the parameters of a Gaussian (μ and σ^2) cannot be applied. In principle, the desire is to obtain a set of parameters that maximize the likelihood of the data. The problem can therefore be solved by applying a special case of the Expectation-Maximization algorithm [Moo96] where values for the mean, variance and mixing coefficient are estimated for each Gaussian component.

An outline of the full procedure for learning a TNBN using the LIPS algorithm is provided in Algorithm 3.1. Subsequently each of the steps is discussed.

Discretizing the temporal data: Initially, the data from which a TNBN is to be built contains temporal data expressed as continuous values represented with real numbers, $\mathbb{R}$. The algorithm begins by performing a discretization of the continuous data with a clustering

Algorithm 3.2 The Initial Interval Approximation Algorithm

Require: A set of ordered centroids P_i, the temporal data D for a TN
Ensure: A set of intervals I
  $min = minimum(D)$
  $max = maximum(D)$
  $I[0].lower = min$
  $I[0].upper = average(P_i[0], P_i[1])$
  for $i = 0$ **to** $size(P_i) - 2$ **do**
    $I[i+1].lower = average(P_i[i], P_i[i+1])$
    $I[i+1].upper = average(P_i[i+1], P_i[i+2])$
  end for
  $i = size(P_i) - 1$
  $I[i].lower = average(P_i[i-1], P_i[i])$
  $I[i].upper = max$
  return I

algorithm. For example, k-means. The result of applying k-means is a set of n data points that represent the centroids of the learned clusters. The clusters are then transformed into intervals with Algorithm 3.2. This algorithm receives as input an ordered set (ascending order) of centroids and the temporal data for the TN. It begins by identifying the minimum and maximum values for the temporal data, and subsequently, it builds the first interval by assigning the lower bounds of the interval to be the minimum value of the temporal data, and the upper bounds to be the average between the first two centroids. The algorithm then enters a cycle where it learns a set of $n-2$ intervals by assigning as lower and upper bounds, the averages of two consecutive centroids. Finally, the last interval is obtained by assigning the lower bounds of the interval to be the average between the last two centroids, and the upper bounds to be the maximum value of the temporal data.

Learning the structure: Once the continuous data has been discretized, the K2 algorithm is applied to learn the structure.

Refining the intervals: In this step, information about the structure of the model is

used to obtain better interval sets. Temporal nodes are addressed in a top down fashion, where the possible configurations for the parents of a TN are obtained. For example, in Figure 3.1 the TN "Coma" would have 4 parental configurations: $\{SC = true, BT = true\}$, $\{SC = true, BT = false\}$, $\{SC = false, BT = true\}$ and $\{SC = false, BT = false\}$. Subsequently, the data set is split up in to several smaller sets, one for each configuration. Configurations with an amount of examples lower than the threshold β provided in Equation 3.2 are discarded, as it is considered that these configurations do not hold enough information to learn meaningful intervals.

$$\beta = \frac{Number\,of\,examples\,in\,the\,configuration}{Number\,of\,configurations \times 2} \tag{3.2}$$

With the resulting configurations, a number of 1 to ℓ Gaussian mixture models are built using each of the data sets obtaining as a result different sets of clusters that correspond to the temporal intervals. To obtain the parameters of the GMM the EM algorithm is applied using a parameter of ℓ, specifying the number of Gaussian distributions for which parameters are being estimated. The interval sets obtained for a configuration are then combined with all of the sets obtained for the rest of the configurations. As a result, different possible interval sets are generated. Figure 3.3 shows a graphic description of this procedure. The resulting intervals sets need to be adjusted so no gaps are present between intervals, and the entire temporal range is covered. Algorithm 3.3 shows the procedure used for adjusting the bounds of each interval in the set. First, intervals are ordered by their lower bounds. Subsequently, if any interval is contained in another interval, both intervals are replaced by a new interval which is an average of the two. This new interval is obtained by defining the lower bounds to be the average of the lower bounds for the two previous intervals, and the upper bounds to be the average of the upper bounds for the two previous intervals. Possible gaps between adjacent intervals are eliminated, such that the upper bounds for an interval is continuous with the lower bounds of the next interval. For this, an average of the lower and upper bounds of two adjacent intervals is obtained, replacing both values.

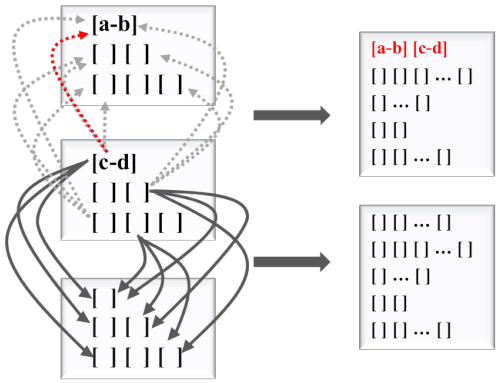


Figure 3.3: A graphic description of the combination process for the intervals sets obtained for a configuration. In this example each configuration generates 3 Gaussian mixture models ($\ell = 3$) which are then combined with the GMMs generated by other configurations.

Algorithm 3.3 The Bounds Adjustment Algorithm

Require: A set of intervals **I**
Ensure: A set of adjusted intervals I'
  for $I \in$ **I do**
    Order I by lower bounds
    while $i \in I$ is contained in interval $i+1 \in I$ **do**
      Replace intervals i and $i+1$ with an average of both intervals
    end while
    for $j = 1$ **to** $|I|$ **do**
      Interval[j].upperBounds=avg(Interval[j].upperBounds,Interval[j+1].lowerBounds)
      Interval[j+1].lowerBounds=avg(Interval[j].upperBounds,Interval[j+1].lowerBounds)
    end for
  end for

Finally, of these possible sets, only those with more than one interval and less than five intervals are considered in order to reduce the network complexity. The best in terms of predictive precision is selected, where the predictive precision is evaluated with the Relative Brier Score [Bri50]. This measure gives an evaluation of the entire model by instantiating a random set of nodes, and then inferring the remaining hidden nodes. The relative Brier Score expressed as a percentage is defined as:

$$RBS = \left(1 - \frac{1}{n}\sum_{i=1}^{n}(1 - P_i)^2\right) \times 100 \tag{3.3}$$

where n is the number of selected nodes to infer, and P_i is the marginal posterior probability of the correct value for each node given the evidence.

Learning the parameters: When refining the intervals, the best set of intervals is selected in terms of predictive precision. This implies that the parameters for the model are calculated when refining the intervals (since the predictive precision could not be measured otherwise). To learn the parameters of the model a maximum likelihood approach is applied. For this step, it is assumed that sufficient data is available to learn a set of parameters close to the true probability distribution.

The LIPS algorithm iterates between learning the structure and refining the intervals. The assumption is that the newly obtained intervals will improve the resulting structure (and vice versa), until a convergence is the structure is reached. In the next section an example of the LIPS algorithm is provided.

Example

In this section an example of the LIPS algorithm, with one iteration, is shown using the network displayed in Figure 3.1. Assume that a set of N examples are available, and that these examples are all in the form shown in Table 3.1. The process begins by discretizing the continuous data by means of a clustering algorithm. In this example two temporal nodes are present, so the clustering algorithm is applied to data data of each continuous variable. The result is shown in Table 3.2.

Once the data has been discretized, the K2 algorithm is applied to learn the structure. The K2 algorithm requires a ordering on the nodes to be given. Since no temporal nodes can be root nodes in a TNBN model, certain restrictions can be made, and thus the number of

Cancer	Serum Calcium	Brain Tumor	Coma	Headache
True	False	False	7	16
True	True	False	13	9
False	True	True	22	39
True	False	True	17	12
False	True	False	41	-
False	False	True	-	-
⋮	⋮	⋮	⋮	⋮

Table 3.1: An example of the data from which a TNBN is learned. The data for temporal nodes ("Coma" and "Headache") is in continuous form representing their day of occurrence, where "-" indicates that the event did not occur.

Cancer	Serum Calcium	Brain Tumor	Coma	Headache
True	False	False	[0-10]	[10-17]
True	True	False	[10-25]	[0-10]
False	True	True	[10-25]	[17-47]
True	False	True	[10-25]	[10-17]
False	True	False	[25-45]	-
False	False	True	-	-
⋮	⋮	⋮	⋮	⋮

Table 3.2: An example of the results of discretizing the data presented in Table 3.1 with a clustering algorithm. The continuous values are replaced with the intervals in which they are contained.

possible orderings reduced. This particular domain allows certain assumptions to be made on the ordering, since it is quite obvious that "Cancer" is the catalyst to all other events, and not the other way around. The following order was defined: Ca, SC, BT, Co, H. This order results in the DAG shown in Figure 3.1.

With the structure of the model defined, the intervals can now be refined. In this example, only the procedure done for the node "Headaches" is shown. This node has as a parent the node "Brain Tumor" which can be either $true$ or $false$. Therefore, two parental configurations are defined, and the N examples of the data set are split up accordingly, such that two data sets are formed. This first data set contains all examples where $BT = true$, while the second

Configuration	Interval Sets	
$BT = true$	$\ell = 1$	$[9 - 38]$
	$\ell = 2$	$[9 - 17][23 - 45]$
	$\ell = 3$	$[5 - 12][15 - 23][27 - 43]$
$BT = false$	$\ell = 1$	$[3 - 40]$
	$\ell = 2$	$[7 - 20][27 - 40]$
	$\ell = 3$	$[3 - 15][17 - 29][33 - 46]$

Table 3.3: The interval sets obtained for the node "Headache" for each parental configuration using the EM algorithm to estimate the parameters for 1 to 3 Gaussian mixture models

Interval Sets for "Headache"
$[7 - 14][15 - 32][33 - 40]$
$[3 - 14][15 - 33][34 - 46]$
$[6 - 26][27 - 45]$
$[8 - 22][23 - 42]$
$[9 - 26][27 - 43]$
$[4 - 15][16 - 26][27 - 38][39 - 46]$

Table 3.4: Possible interval sets for the TN "Headache"

contains all examples where $BT = false$. A number of 1 to $\ell = 3$ GMMs are obtained by applying the EM algorithm to learn the parameters, resulting in a total of 6 interval sets for both configurations. Note that ℓ is a user-defined value that will determine the number of the interval sets. In order to keep complexity low, this value is set to be 3. Table 3.3 shows the intervals calculated with the GMM model.

The interval sets for one configuration are then combined with the intervals sets for the other configuration, resulting in $3 \times 3 = 9$ possible sets; however, some of these sets are not considered as they hold too few intervals or too many. For example, consider the first combination $[9 - 38]$ and $[3 - 40]$. First, the intervals are ordered by their lower bounds, resulting in $[3 - 40][9 - 38]$. Subsequently, the algorithm verifies if one interval is contained in the other. In this case, interval $[9 - 38]$ is contained in $[3 - 40]$, and consequently an average interval is calculated, resulting in the new interval $[6 - 39]$. The interval set resulting from this combination would be formed solely by the interval $[6 - 39]$. This interval set is

discarded because it does not meet the minimum number of intervals required for a TN.

Next, consider the combination $[9-38]$ and $[7-20][27-40]$. Again, intervals are ordered by their lower bounds, resulting in $[7-20][9-38][27-40]$. In this case, no interval is contained in another, and thus the algorithm only needs to adjust the bounds of each interval. The resulting interval set is $[7-14][15-32][33-40]$. This combination process is repeated for all interval sets, obtaining as a result the intervals displayed in Table 3.4. Of the resulting sets, the best in terms of predictive accuracy is selected. This same process is repeated for the remaining TN, "Coma".

3.3 Chapter Summary

In this chapter the Temporal Nodes Bayesian Network model was presented. The TNBN model is the focus of the present thesis. An algorithm for learning a TNBN was introduced, where a novel strategy for learning the temporal intervals of a temporal node was defined by the authors. This algorithm, denominated "Learning Intervals, Parameters and Structure" or LIPS, assumes that sufficient data is available for learning, and in situations where data is scarce, unreliable models will be obtained. In dynamic domains, data can become outdated very quickly, or it may be difficult to be constantly collecting new samples. Without a sizable data set traditional machine learning algorithms will produce models with adverse effects. In the next chapter, a strategy for learning reliable models when data is scarce is presented. Applying this strategy for learning a TNBN when the available data is limited is the primary objective of this thesis.

Chapter 4

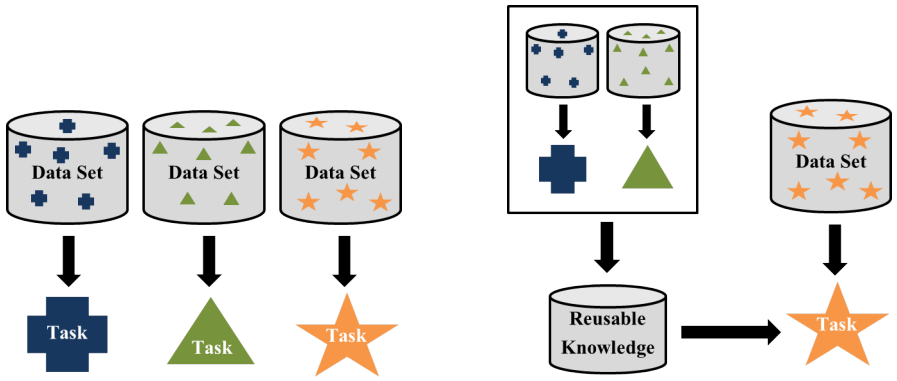
Transfer Learning

In this chapter transfer learning, a strategy for learning models by reusing what has already be learned for other tasks, is presented. First, a brief introduction, along with a formal definition of transfer learning is provided. In the next section, a general overview of domains where transfer learning has been applied is presented. This section is followed by a brief discussion about the open challenges that remain for transfer learning. Finally, the discussion is shifted towards the state of the art for transfer learning with probabilistic graphical models, giving special emphasis on the work done by Luis et al. in [LSM10], as the algorithm proposed in this thesis draws on their work.

4.1 Formal Definition

Traditional machine learning algorithms build a model from a set of examples unique to a specific domain. In contrast to how machines acquire knowledge, human beings reuse what they have already learned from previous experiences when presented with a new learning task. In effect, they transfer knowledge by recognizing similarities between the task they want to learn and the tasks they have previously learned. An example could be made with how toddlers learn to run by reapplying the information they acquired when learning to walk.

Transfer learning or knowledge transfer is machine learning that like human learning,



(a) Traditional machine learning paradigm

(b) Transfer learning paradigm

Figure 4.1: Illustration of the differences between the two learning paradigms. Traditional machine learning algorithms (a) build a model from domain specific examples, whereas transfer learning (b) extracts useful knowledge from what has already been learned to build the model for a new task.

reapplies the knowledge acquired from one or more auxiliary tasks for training a new similar task. This reuse of knowledge can make the transfer learning paradigm much more efficient than the traditional paradigm. Specifically, it can improve the generality of the learned models, and the speed with which a particular algorithm converges to a reliable model. In addition, traditional machine learning algorithms rely heavily on the amount of data available for a given domain. These algorithms assume there is sufficient information to obtain an accurate model, and in situations where data is scarce, the resulting models may be unreliable. Transfer learning can overcome this obstacle by learning the model from several sources, thus compensating for a scarce data set. Figure 4.1 illustrates the differences in the traditional machine learning paradigm and the transfer learning paradigm.

Two important concepts in transfer learning are "domain" and "task". A domain consists of a feature space X and a probability distribution $P(X)$ over X. A feature space defines the variables present in the task of interest. Two domains are said to be dissimilar when they have different feature spaces or their probability distributions are not the same. Given a specific domain, a task is a model that is learned from a set of examples belonging to the

domain. This model is a predictive function that gives a general description of the domain, and allows predictions to be made for new unseen examples.

Transfer learning allows domains, tasks and distributions to be different when training and testing. This allows the constraints on where transferred knowledge can come from to be less restrictive. A formal definition for transfer learning was provided by Pan and Yang in [PY10].

Definition 4.1. *Given an auxiliary domain D_A and a learning task T_A, a target domain D_T and a learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_A and T_A, where $D_A \neq D_T$, or $T_A \neq T_T$.*

4.2 Application Domains

Recently, the interest in algorithms that apply transfer learning has increased because they eliminate the need to be constantly collecting and labeling new data; a task that can often be expensive and in some cases take too long.

Transfer learning has been previously used in areas such as document classification. In document classification the objective is to assign a given document to one or more classes or categories. The assignment is typically done based on the content of the document, such that it is the subjects discussed in a document that determine the category or categories to which it belongs. Yang et al. applied transfer learning to build a classification model for a target class with no labeled training examples [YJJ+10]. In their work, they use the generalized maximum entropy model to learn a binary classifier where the label information from a set of auxiliary classes are transferred to the target class. Similarly, Dai et al. proposed the co-clustering based classification (CoCC) algorithm to classify a set of out-of-domain documents by transferring labels [DXYY07a]. Unlike the work of Yang et al. the CoCC algorithm does not learn a model for a new class, rather it aims to assign the class labels from the in-domain

documents to the out-of-domain documents by identifying common word clusters between the domains.

Web applications have also benefited from transfer learning. Similar to the text classification problem, one common challenge in this domain is to be able to successfully classify a Web page based on its content. While effective work has been done for Web pages that are all in the same widespread language (e.g. English), Web pages with content written in less common languages have a smaller amount of labeled examples, and thus classifying them poses a more challenging problem. Wang et al. propose an approach for transferring knowledge between Web pages of different languages to address this classification problem [WHND11]. By recognizing similarities in the semantic patterns of the cross-language contents, they were able to improve the accuracy with which Web pages in less widespread languages were classified.

Transfer learning has also been used for Recommender Systems (RS). Recommender systems employ the user's historical data to offer suggestions on other content that might be of the user's liking. Traditionally, most RS work in a single domain (e.g. books, music, movies, etc.), however Moreno et al. use a transfer learning technique to learn a cross-domain RS [MSRS12]. By transferring knowledge from multiple domains (e.g. movies and music), they can generate suggestions for another target domain (e.g. games).

Image and video domains have also found applications where transferring knowledge is beneficial. Zhu et al. transfer information between text and images to improve an image classifier [ZCL+11]. They take advantage of the existing annotated images found on diverse Web sites and use this information to relate the images to the large amounts of documents available online. They then leverage this knowledge to classify a new image.

Transfer learning has also been applied in the area of video surveillance. Wang et. al propose a strategy for transferring a generic pedestrian detector to a specific scene [WLW12]. While a generic pedestrian detector would not work well when applied to a scene outside of its training set, Wang et al. suggest that it is possible to use the generic model as a starting point, and automatically train it for a specific scene. Their approach achieved significant

improvements over the generic model for two data sets corresponding to separate scenes.

Works that use reinforcement learning have also benefited from transferring knowledge. In [TSL05] the authors use the final value function of a source task as the initial solution for the target task. This strategy is called a starting-point method, and it can significantly speed up the time it takes for the algorithm to reach a good solution for the target task.

4.3 Open Challenges

While the goal of transfer learning is to improve the learning for a new task, situations may occur where the performance instead decreases. For these cases, it is said that *negative transfer* has occurred. Transfer learning can become detrimental if the auxiliary sources of transfer are not sufficiently related to the target task. Avoiding negative transfer is one of the major challenges currently facing transfer learning.

To achieve a positive effect from transfer learning, two aspects must be considered:

- The sources of transfer must maintain a close relationship to the target task
- The transfer method must correctly leverage the pertinent information

The effectiveness of any transfer learning method is contingent on the relationship between the target task and the auxiliary sources of transfer. A strong relationship implies that the two tasks are closely related, and thus there are many similarities which transfer learning can use favorably. Figure 4.2 illustrates the association between strong task similarities and positive transfer. As tasks become more similar, the possibility of positive transfer also becomes stronger. It is at its strongest when tasks are identical, and in this case there is no need for transfer, as a model has already been learned (the auxiliary model). As tasks become more dissimilar, positive transfer learning is more difficult to achieve culminating with utterly disparate tasks where only negative transfer is possible.

Figure 4.2: A representation of how task similarity relates to the effectiveness of transfer learning. Transfer learning is more effective when tasks have a strong relationship, reaching its climax when both tasks are identical. As tasks become more dissimilar positive transfer and can become negative transfer if task similarity is insufficient or nonexistent as is the case with completely disparate tasks.

Even if a strong relationship is observed, it is not necessarily true that all the information from the auxiliary source is relevant. Recognizing which pieces of information to use should be an important aspect in any transfer learning method.

To identify the auxiliary sources of information most similar to the target task it is possible to explicitly model the relationships between tasks. Because in many cases, a transfer learning method is an extension of a machine learning algorithm, how the relationships are modeled will be highly dependent on the algorithm being used. A good evaluation of the relatedness of two tasks can minimize (or avoid altogether) the effects of negative transfer; however, being overly cautious may also reduce the benefits of positive transfer.

Another challenge facing transfer learning is *mapping*. While two tasks may be similar, knowledge transfer can only occur if they are both represented in the same manner. For example, two systems may express similar behaviors, and thus it would be desirable to apply transfer learning to leverage these similarities. However, if the variables used to model the two systems are different, then knowledge transfer cannot occur unless a mapping between properties is achieved. In many cases a domain expert may be able to provide this correspondence; however, strategies to achieve a mapping automatically have also been proposed [PKY08, TKS08, SS06]. In this work it is assumed that all auxiliary tasks have the same properties or at least a subset of the properties of the target task, and therefore no mapping procedure is required.

4.4 Transfer Learning for Probabilistic Graphical Models

Information about a problem often comes incorrect and incomplete, leading to uncertainty. Probabilistic graphical models are commonly used to represent domains where uncertainty is present; however, learning them with insufficient data leads to the model presenting adverse effects and as a result being unreliable. Several strategies that address this problem have already been proposed; for probabilistic graphical models the focus is mainly on learning reliable probability distributions and structures when few instances are available for training.

Tonda et al. applied an evolutionary algorithm to induce the structure of a Bayesian network [TLR+12]. The structure is obtained using a score and search based strategy to explore the space of possible graphs, where the candidate graphs are evaluated using a fitness function based on information entropy. Despite having data sets of limited size, the authors report promising results for the structures obtained when training with few instances.

The work of Onisko et al. focuses on the probability distributions for the models [ODW01]. The authors propose a methodology that uses Noisy-OR gates to compensate for lack of information. In this work the structure of the Bayesian network is assumed to be known, and therefore only the parameters require learning.

Transfer learning strategies have also been proposed to learn the components of Bayesian networks. Dai et al. attempted to learn the parameters of a naive Bayes network used for text classification [DXYY07b]. Initial parameters were estimated using labeled data, and then these were refined with the help of an EM-based algorithm that introduced knowledge from a separate data set belonging to a different distribution.

Niculescu-Mizil and Caruana propose an algorithm for learning the structure of various Bayesian Networks by transferring knowledge between similar tasks [NMC07]. This simultaneous learning receives the name of *multitask learning*, and it differs slightly from typical transfer learning in that more than one target task exists, and each target task is also a source task. In their work, the authors use a greedy heuristic search algorithm to learn the best set of structures for a set of tasks, and they work under the assumption that all tasks

are related. Evaluation is done with a heuristic that penalizes structures that deviate significantly from each other, so that the result is a set of structures that are relatively similar to each other.

4.4.1 Inductive Transfer for Bayesian Networks

While efforts have been made to compensate for lack of training information, most of these focus solely on one aspect of the model i.e., the structure or the parameters. Luis et al. propose a transfer learning strategy to learn both components of a Bayesian network [LSM10]. To learn the structure they proposed the PC-TL algorithm, which is an extension of the PC algorithm for transferring knowledge between tasks. A general outline of the algorithm is provided in Algorithm 4.1. PC-TL follows the same procedure as the PC algorithm; however it changes how the independence tests are evaluated. In the PC algorithm, an independence test evaluates whether two variables are independent given another variable, or an other set of variables. If two variables are determined to be independent, no arc will exist between them in the Bayesian network. In PC-TL the independence measure is a linear weighted combination of the independence tests performed on the target data with the independence tests performed on the most similar auxiliary task. The combined independence measure is given by the following function:

$$I_F(X,Y|S) = \alpha_0(X,Y|S) \times sgn(I_0(X,Y|S)) + Sk_{XY} \times (\alpha_{D_{XY}}(X,Y|S) \times sgn(I_{D_{XY}}(X,Y|S))) \quad (4.1)$$

where $sgn(I)$ is -1 if X, Y are independent given S, and $+1$ otherwise. $\alpha_0(X,Y|S)$ and $\alpha_{D_{XY}}(X,Y|S)$ are confidence measures for the target and the most similar auxiliary task respectively, and Sk_{XY} is a similarity metric. The confidence and similarity metrics are further discussed below.

Algorithm 4.1 The PC-TL Algorithm

Require: Set of variables X, Independence test I
Require: Target data D_0, Auxiliary data $D_1, \ldots, D_m$
Ensure: Directed Acyclic Graph G
  Initialize a completely connected undirected graph G'
  $i = 0$
  repeat
    for $X \in \mathbf{X}$ **do**
      for $Y \in ADJ(X) - \{Y\}$, $|S| = i$ **do**
        Find the most similar auxiliary task k and its similarity measure Sk_{XY}
        Determine the confidence measures $\alpha(X, Y|S)$ for the target and auxiliary tasks
        Obtain the combined independence measure $I_F(X, Y|S)$
        if $I_F(X, Y|S)$ **then**
          Remove the arc $X - Y$ from G'
        end if
      end for
    end for
    $i = i + 1$
  until $|ADJ(X)| \leq i, \forall X$
  G =Orient arcs in G'
  return G

The Confidence Metric

The value $\alpha(X, Y|S)$ is a measure of the confidence in the performed independence tests. One way to evaluate independence between variables is the conditional cross entropy metric. This metric evaluates the mutual information between two variables X and Y given a third variable, or variable set Z; it is defined as:

$$CE(X, Y|Z) = \sum_{z} P(z) \sum_{x,y} P(x, y|z) \log \frac{P(x, y|z)}{P(x|z)P(y|z)} \tag{4.2}$$

The conditional cross entropy metric is dependent on the size of the dataset. Based on the empirical evaluation done by Friedman and Yakhini in [FY96], which shows that the error of this test is asymptotically proportional to $\frac{\log N}{2N}$, where N is the size of the dataset, Luis et al. defined the following function to determine the confidence of the independency

test between two variables X and Y given the conditioning set S.

$$\alpha\left(X, Y|S\right) = 1 - \frac{\log N}{2N} \times T \tag{4.3}$$

where $T = |X| \times |Y| \times |S|$, and $|W|$ is the cardinality of W. If this value becomes negative α is set to 0.005.

The Similarity Metric

In order to transfer from tasks that are more closely related with the target task, Luis et al. measure the similarity between the auxiliary tasks and the target task, transferring only from the most similar task. The similarity metric Sk_{XY} is an evaluation of both local and global similarity, and it is defined in terms of the number of shared independencies between the target task and an auxiliary task. As previously mentioned, modeling the strength of the relationship between tasks is dependent on the machine learning algorithm being used. Because PC-TL is an extension of the PC algorithm it is appropriate to use the number of shared independencies to evaluate relatedness.

Global similarity is measured as follows:

$$Sg_{Dj} = dep_j + ind_j \tag{4.4}$$

where dep_j is the number of common conditional dependencies between all variable pairs in the target task and the j^{th} auxiliary task, and ind_j is the number of common conditional independencies between all variable pairs in the target task and the j^{th} auxiliary task. Conditional (in)dependence is evaluated with the conditional cross entropy measure using a threshold.

Local similarity for two variables X, Y given a conditioning subset S is defined as:

$$Sl_{Dj}(X,Y) = \begin{cases} 1.0 & if\ I_0(X,Y|S) = I_{Dj}(X,Y|S) \\ 0.5 & if\ I_0(X,Y|S) \neq I_{Dj}(X,Y|S) \end{cases} \tag{4.5}$$

where $I_0(X,Y|S)$ is a true or false value indicating independence (true) or dependence (false) between X and Y in the target task, and $I_{Dj}(X,Y|S)$ is a true or false value indicating independence or dependence between X and Y in the auxiliary task. The constants 1.0 and 0.5 are a weight for auxiliary tasks that have the same or different local structures. Auxiliary tasks with the same local structure are preferred, and therefore more weight is given to tasks of this type.

The similarity metric is a combination of the global and local evaluation. Defined for the most similar auxiliary task k as:

$$Sk_{XY} = Sg_{Dk} \times Sl_{Dk}(X,Y) \tag{4.6}$$

Learning the Parameters with Transfer Learning

For the obtained structure of the Bayesian network Luis et al. learned the parameters of the model by applying a transfer learning strategy that combines conditional probability tables from several sources using linear aggregation also known as weighted mean. To combine the conditional probability tables for a specific node, the node in the auxiliary task must have the same parents as its counterpart in the target task, and if this is not the case the auxiliary structure must be altered. Three cases are considered:

1. The node in the auxiliary structure has more parents. In this case the additional parents are removed by marginalizing over them to obtain the desired substructure.

2. The node in the auxiliary structure has less parents. Here, values for the additional parents are obtained by repeating the values seen in the auxiliary conditional proba-

bility table for the common parents. For example, the values of $P(X|Y)$ are repeated for all values of Z in $P(X|Y,Z)$.

3. A combination of 1 and 2. In this case additional parents are first marginalized over, and then the procedure done in case 2 is performed.

Once all auxiliary and target conditional probability tables have the same structure, these tables are combined using a weighted mean as follows:

$$P(X) = k \times \sum_{i=1}^{n} w_i P_i(X) \tag{4.7}$$

where $P_i(X)$ represents the conditional probability of the i^{th} task for a variable X, w_i is a weight associated to that probability, and k is a normalizing factor.

4.5 Chapter Summary

In this chapter the transfer learning paradigm was presented. Several works done for domains that have benefited from applying transfer learning were presented, and while this included efforts in the area of probabilistic graphical models, it did not include models that consider dynamic information. To the knowledge of the author, there are no existent strategies that use transfer learning to build dynamic Bayesian models. Unfortunately, in dynamic domains, data can become outdated very quickly, or collecting it may take an unfeasible amount of time resulting in scarce data sets. In these cases, a transfer learning strategy that could compensate for the lack of information would be desirable. This thesis proposes a methodology for learning a Temporal Nodes Bayesian Network with transfer learning. In the next chapter, a detailed discussion of the methodology is presented.

Chapter 5

Transfer Learning for Temporal Nodes Bayesian Networks

In this chapter the proposed methodology for inducing a Temporal Nodes Bayesian Network with transfer learning is presented. A procedure for learning each component of a TNBN is defined and discussed, where knowledge is transferred from auxiliary source domains in order to compensate for the lack of target information. An example follows the discussion of the transfer learning procedure for each component, so that a better understanding of the proposed algorithm can be achieved.

5.1 TNBN-TL

A TNBN consists of three components: 1) the structure, 2) the intervals in which temporal events occur, and 3) the probability distributions that parameterize the model. To fully learn a TNBN, a procedure for learning each of these elements must be defined. Accordingly, learning a TNBN with knowledge transfer consists in applying transfer learning to the procedures for inducing each component. In order to transfer knowledge, *how* and *what* to transfer must be defined. A basic outline of the methodology for learning a TNBN with

Algorithm 5.1 TNBN-TL

Ensure: TNBN
InitialInterval_TL();
Structural_TL();
IntervalRefinement_TL();
Parametric_TL();

transfer learning is provided in Algorithm 5.1, and graphic description is presented in Figure 5.1. In the next subsections each of these steps is discussed in detail.

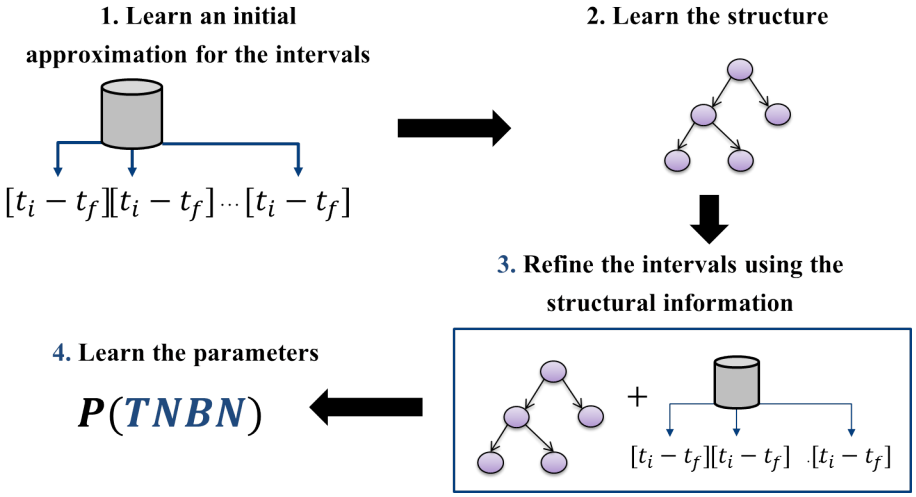


Figure 5.1: A graphic overview of the TNBN-TL methodology.

5.1.1 Learning the Initial Intervals with Transfer Learning

Initially, the temporal data present in the target domain is expressed as continuous values just as shown in Table 3.1 for the metastatic cancer TNBN example. Before a structure for the model can be learned, the continuous data must be discretized. This corresponds to finding an initial approximation of the temporal intervals for each temporal node. A strategy similar to the one used by Hernandez-Leal et al. in [HLGMES13] is applied, where the k-means algorithm is used to obtain a set of clusters corresponding to the temporal intervals of a TN. It is important to mention that initially the number of intervals for each TN (i.e.,

the value of k) is a predetermined number; however further on it is shown how to induce the number of intervals that each TN has. For the experiments carried out in this thesis, k is set to be 3; the mid value between the minimum number of intervals possible (i.e., 2) and the maximum allowed (i.e., 4) in order to keep network complexity low.

Since the data available for the target domain is scarce, the temporal intervals obtained using only this small data set may be unreliable. By transferring data between a TN in the target domain, and its counterpart in an auxiliary domain, a better approximation of the intervals for each TN can be obtained. More specifically, the intention is to add auxiliary temporal information to the data on which k-means will be applied, and in order to do this, temporal data from the auxiliary source domains is generated to augment the scarce data set. It is assumed that the original continuous data for the auxiliary domains is not available; if it were available, this data could be used, thus eliminating the need for data generation.

The basic procedure for learning a set of temporal intervals for a TN with knowledge transfer is illustrated in Figure 5.2. This procedure makes the assumption that each interval is characterized by a Gaussian distribution (step 1), where μ is the middle point of the interval and σ is the distance from that point to either of its lower or upper limits. Under this assumption it is possible to generate random numerical values that follow the Gaussian distribution parameterized by μ and σ, and that are constricted to the range $\mu \pm \sigma$. To incorporate data from an auxiliary domain to the interval learning process, a set of continuous values from all the intervals belonging to the TN is generated (step 2). Each interval generates data following the TN's probability distribution. This set is then added to the target data, and finally k-means is applied (step 3), where each cluster corresponds to a temporal interval.

It is possible that the data generated from auxiliary sources may completely overwhelm the scarce amount of target data, thus increasing the likelihood of negative transfer. To avoid such a scenario two safety measures are taken. First, the amount of transferred auxiliary data is controlled based on its similarity to the target domain; and second, the amount of target data is increased by applying the same procedure described in Figure 5.2.

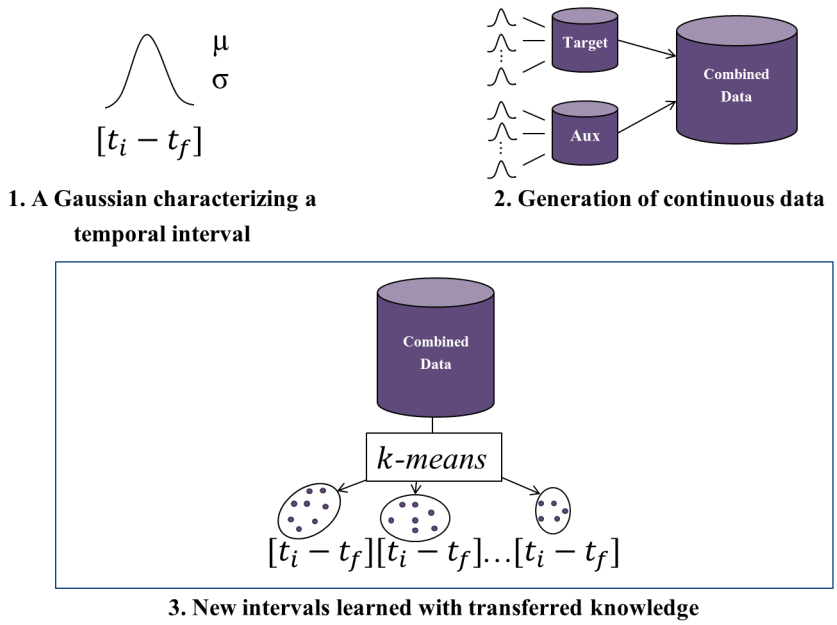


Figure 5.2: Illustration of the procedure followed for obtaining an initial approximation of the temporal intervals for a temporal node.

Controlling the Amount of Transferred Auxiliary Data

In order to avoid the auxiliary information from overwhelming the target data, it is necessary to decide how much data each auxiliary domain transfers, and to do this, the size of the combined data set must be established beforehand. To give a good estimation of the total amount of continuous records needed to learn the intervals for each TN, the size of the combined data set is established to be the same as that of the data set belonging to the auxiliary domain most similar to the target domain. Since each auxiliary domain is presumed to be learned with sufficient data, it is assumed that the amount of data necessary to learn a reliable target model is close to the amount of records used for the most similar domain. This strategy assumes that some knowledge on the sizes of the auxiliary data sets is available. An alternative, is to estimate the amount of data necessary to calculate all the probability tables by approximating it to 10 records for each probability value present in the

model.

For determining the auxiliary source most similar to the target domain, the global similarity metric defined in Equation 4.4 is used to evaluate the strength of the relation between an auxiliary domain and the target domain. In other words, similarity is measured by the number of shared dependencies and independencies, where two variables are said to be independent if their conditional cross entropy measure is below a certain threshold.

Once the size of the combined data set has been determined, it is necessary to decide how much each auxiliary domain is going to transfer, that is, how many records they are going to contribute to the total. This amount is decided by how similar that auxiliary domain is to the target domain using Equation 4.4. The reasoning behind this is that it would be preferable for domains closer to our target domain to contribute with more records than those domains that are less similar.

Increasing the Amount of Target Data

In addition to controlling the proportion of auxiliary records, it is also possible to increase the amount of target data by applying the same process as for the auxiliary source domains. To obtain initial temporal intervals from which continuous values can be generated, k-means is applied to the continuous target data for the temporal node of interest. The process applied on the auxiliary domains is then repeated in order to increase the proportion of target information in the total amount of records. Finally, k-means is applied to the combined data set of target and auxiliary records to obtain k intervals learned with knowledge transfer. For the experiments carried out in this thesis the target domain accounts for 25% of the total data.

Algorithm 5.2 is an outline of the basic steps previously discussed for learning the temporal intervals of a TN with transfer learning. For every auxiliary TN corresponding to a counterpart to the target TN, the process begins by calculating how much the auxiliary source contributes to the total amount of data N. Based on this number, continuous records

Algorithm 5.2 Interval-TL for a TN

Require: TN_{aux} /* A set of the TN's counterparts in the auxiliary domains */
Require: D /* A data set containing the target data for the TN */
Require: N /* The amount of records to generate */
Require: k /* The number of intervals */
Ensure: I /* A set of intervals */
  for all $tn \in TN_{aux}$ **do**
    $M \leftarrow calculateProportion(tn.similarity, N)$; /* Calculates the proportion of N the auxiliary domain generates based on its similarity to the target domain */
    $data \leftarrow generateData(tn, M)$;
    $D.add(data)$;
  end for
  $I \leftarrow k$-means(k, D);
  return I

are generated for the auxiliary source and added to a database which combines all the records. Finally, k-means is applied to this new data set to determine the temporal intervals.

Example of Transfer Learning for Initial Interval Approximation

An example of how to induce an initial approximation of the intervals for a temporal node is now presented. Consider the TN "Headaches" for the TNBN presented in Figure 5.3. Initially, the data for the target domain is in continuous form as shown in Table 5.1.

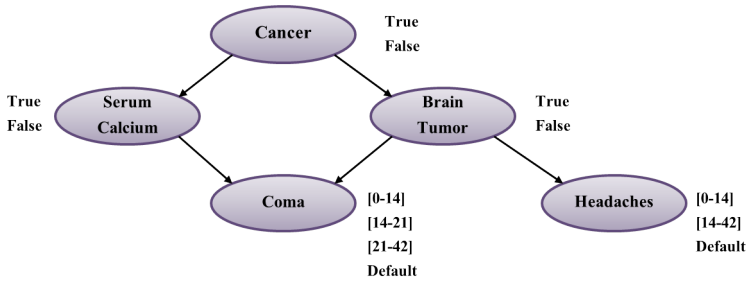


Figure 5.3: A temporal nodes Bayesian network modeling the outcomes of being diagnosed with metastatic cancer. "Cancer", "Serum Calcium" and "Brain Tumor" are instantaneous nodes, and "Coma" and "Headaches" are temporal nodes each with a set of intervals and a default value.

Headache (day of occurrence)
16
9
39
12
-
-
⋮

Table 5.1: Continuous data for the temporal node "Headache" indicating the day the event takes place. The value "-" indicates that the event did not take place.

Now consider an auxiliary source model built from 500 records, and with a TN "Headaches" with temporal intervals "[0 - 10]", "[10 - 30]", "[30 - 45]", and a "Default" value indicating that a headache did not occur. The TN "Headaches" for this auxiliary source is presented in Figure 5.4.

In order to transfer information from the auxiliary to target source, first the amount of data to be generated must be calculated. In this case, because only one auxiliary source exists, this amount is set to be the size of the data set for the auxiliary source, i.e., 500 records. The next step is to determine how much the auxiliary source will contribute to the total amount of records. This value is calculated by simply subtracting the amount of records that the target source will contribute from the total amount of records. If the target source contributes 25% then the auxiliary source domain will contribute 75%, that is, 375 records.

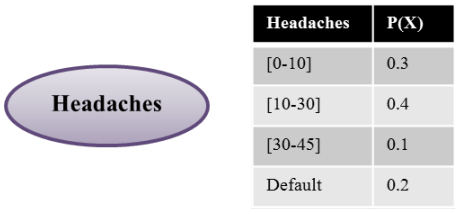


Figure 5.4: A temporal node representing the variable "Headache" in an auxiliary source domain. The marginal probability distribution for "Headaches" is also shown.

Headache (day of occurrence)
[12-33]
[0-12]
[33-45]
[12-33]
Default
Default
⋮

Table 5.2: Continuous data for the temporal node "Headache" indicating the day the event takes place. The value "-" indicates that the event did not take place.

To create the 375 record data set, continuous values from all the intervals belonging to the auxiliary TN are generated by assuming a Gaussian distribution for each interval. Each temporal interval generates data in the proportion described by its probability distribution, normalizing over the "Default" value. For example, consider the interval "$[0-10]$". Before normalizing over the "Default" value, it has a probability of 0.3; however, after normalizing this value increases to 0.375. Therefore, the interval "$[0-10]$" will account for 37.5% (approximately 140 records) of the data generated by the auxiliary source.

In order to generate the 140 records from the interval "$[0-10]$" it is assumed that a Gaussian distribution characterizes the interval. For this interval μ is 5 (the middle point of the interval) and σ is 5 (the distance from the middle point to either extreme). With these parameters, 140 continuous values that fall in the range $\mu \pm \sigma$ can be generated.

This process is repeated for the remaining temporal intervals to create the full 375 records. Finally, this data is combined with data from the target domain, and the k-means clustering algorithm is applied to obtain an initial approximation of the intervals for the TN "Headache". Table 5.2 shows the updated data for the TN "Headache". Continuous values are now replaced with the intervals obtained from the combined data set, using a value of $k = 3$.

5.1.2 Learning the Structure with Transfer Learning

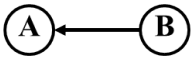
Once a set of initial intervals for each TN is obtained, the structure for the model can be learned from the now discretized data set for the target domain and a set of auxiliary sources. To learn the DAG for the model, the PC-TL algorithm discussed in Chapter 4 is applied. This algorithm transfers knowledge by combining the results of the independence tests performed on the target and auxiliary domains. For transferring between TNBNs, the PC-TL algorithm was modified to use information on the ordering of the nodes when learning the DAG. Since the occurrence of a temporal event requires that a parent event take place, all TNs are child nodes, and therefore all root nodes of a TNBN must be instantaneous. This particular characteristic of the TNBN model allows some restrictions on the ordering of the nodes to be made when learning the structure, thus reducing the search space of possible DAGs. In addition, a partial order can be obtained from the temporal data for the target domain, since it is possible to identify the order of occurrence for some of the temporal events.

To represent the node orderings, an approach that associates to each node a list of possible parents is proposed, where each member of the list is another node in the model. For example, using the model in Figure 3.1, $Brain\,Tumor \leftarrow \{Cancer, Serum\,Calcium\}$ expresses that "Cancer" and "Serum Calcium" are possible parents of the node "Brain Tumor". This form of representation expresses uncertainty in the information present in the list, since members of the list may or may not be parents to the node. Continuing with the previous example, "Serum Calcium" belongs to the list associated to "Brain Tumor"; however, it is known from the model in Figure 5.3 that "Serum Calcium" is not a parent to "Brain Tumor". Additionally, certainty is expressed by omitting a node from the list, i.e., if a node is not a member of the list it cannot be a parent. For example, omitting "Coma" from the list expresses certainty that "Coma" is not a parent to "Brain Tumor".

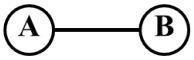
To incorporate this ordering representation into the PC-TL algorithm 3 cases were identified. Assume two nodes A and B belonging to a model, and a fully connected graph G.

1. A is not in B's parent list, and B is not in A's parent list: In this case neither node has the other as a possible parent, and therefore the arc between them in G can be removed. By addressing cases of this type before running PC-TL's main procedure performing unnecessary independence tests is avoided.

2. A is not in B's parent list, but B is in A's parent list (and the inverse): This case expresses that B may be a parent to A. In this scenario, the arc between A and B is attempted to be removed with the independence tests performed in PC-TL. If the arc remains it is oriented as $A \leftarrow B$.



3. A is in B's parent list, and B is in A's parent list: In this case both nodes have each other as possible parents. For this scenario, the arc between A and B is attempted to be removed with the independence tests performed in PC-TL. If the arc remains it is left unoriented.



Algorithm 5.3 gives a general outline of the method. The procedure begins with a fully connected graph. Immediately, arcs that fall within *case 1* are removed to avoid performing unnecessary independence tests. Just as in PC-TL, the algorithm then iterates between variable pairs, and obtains a combined independence measure of the two variables given some subset S of other variables. If two variables are found to be dependent, the arc between them is treated according to which case they fall in (case 2 or 3). If not, it is removed. Finally, unoriented arcs of the graph are directed based on the conditional independence tests performed on variable triplets. Note that before directing the unoriented arcs, this

Algorithm 5.3 PC-TL with node ordering

Require: **X** /* A set of variables */
Require: D_0 /* The target data */
Require: D_1, ..., D_m /* The auxiliary data */
Require: L /* A set of parent lists for each $X \in$ **X** */
Ensure: G /* A DAG */
  Initialize a complete undirected graph G'
  Remove all arcs in G' that fall in *Case 1*.
  for all Variable Pairs $(X, Y) \in$ **X do**
    if $Independent(X, Y \mid S)$ **then**
      Remove arc $X - Y$ from G'
    else
      if $X \in L(Y)$ and $Y \notin L(X)$ **then**
        Orient arc $X \rightarrow Y$
      end if
      if $X \in L(Y)$ and $Y \in L(X)$ **then**
        Leave arc unoriented $X - Y$
      end if
    end if
  end for
  $G \leftarrow$ Orient remaining arcs in G'
  return G

procedure will obtain a partially directed graph. However, if each node has all the remaining nodes associated as possible parents, this situation is equivalent to having *no information* on the causal ordering, and in this case only the skeleton of the structure will be obtained.

It is important to mention that it is possible for arcs, that fall in *case 2*, to form a directed cycle. To prevent the appearance of cycles and to preserve DAG integrity, every time an arc falls in case 2 the graph is tested for cycles. If a cycle has been created, then the arc (belonging to the cycle) with the lowest independence measure is removed, thus breaking the cycle. Precautions are also taken to preserve TNBN integrity by assuring that every TN has at least one parent. This provision is made in the second phase of PC-TL when unoriented arcs are directed.

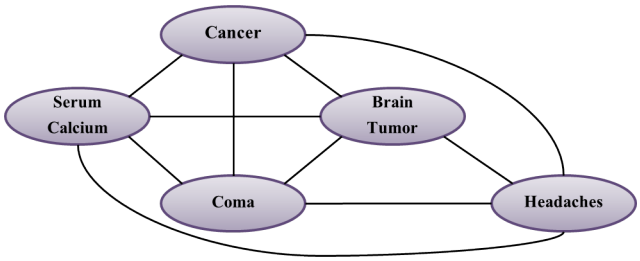


Figure 5.5: The initial fully connected graph from which PC-TL learns the structure for the temporal nodes Bayesian network.

Example of PC-TL with a Node Ordering

An example of the PC-TL algorithm with a node ordering is now presented for the model in Figure 5.3. For this example the following partial order is established:

$Cancer \leftarrow \{Brain\,Tumor\}$
$Serum\,Calcium \leftarrow \{Cancer, Brain\,Tumor, Coma\}$
$Brain\,Tumor \leftarrow \{Cancer, Serum\,Calcium\}$
$Coma \leftarrow \{Serum\,Calcium, Brain\,Tumor\}$
$Headaches \leftarrow \{Serum\,Calcium, Brain\,Tumor\}$

Initially the graph is fully connected, as shown in Figure 5.5. The algorithm begins by removing all arcs between nodes where neither node is in each other's list (case 1). Consider the nodes "Cancer" and "Headaches". The list associated to "Cancer" does not contain "Headaches" as a possible parent. Additionally, "Cancer" is not a member of "Headaches" possible parent list. Since neither node belongs to the other's list, the arc between "Cancer" and "Headache" is removed. Two more arcs fall in this case: the arc between "Cancer" and "Coma", and the arc between "Coma" and "Headaches". Figure 5.6 shows the result of removing all arcs that fall in case 1 from the graph.

Once all arcs that fall in case 1 have been removed, the main procedure for PC-TL is performed. The conditional cross entropy for all variable pairs given a subset S is calcu-

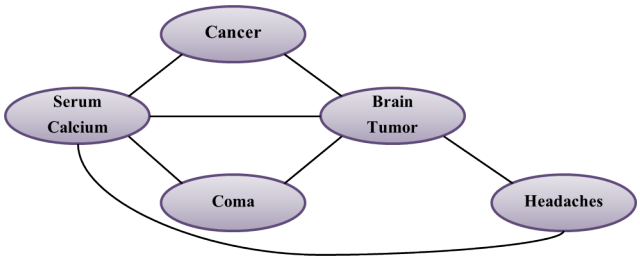


Figure 5.6: The resulting graph of removing the arcs that fall in case 1 from the fully connected graph.

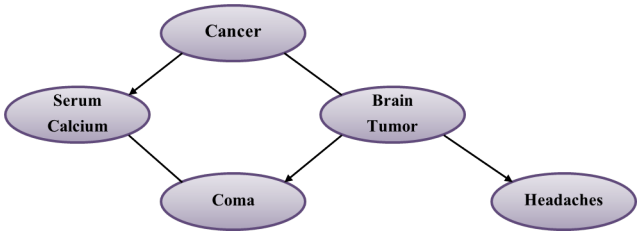


Figure 5.7: The partially directed graph obtained after applying the first phase of the PC-TL algorithm with a node ordering.

lated and combined (as indicated in Equation 4.1) with results for the independence test performed on the auxiliary source. Consider the nodes "Serum Calcium" and "Headaches". The combined independence measure finds that these nodes are independent given a subset of other variables. The arc between "Serum Calcium" and "Headaches" is therefore removed. Conversely, the combined independence measure for the nodes "Cancer" and "Serum Calcium" finds that no subset S exists that makes the two nodes independent. Since "Cancer" is a possible parent of "Serum Calcium" but not vice versa, the arc between the nodes is directed from "Cancer" to "Serum Calcium" (case 2).

Figure 5.7 shows the partially directed graph obtained after executing the main procedure for PC-TL and before unoriented arcs are directed. The arcs between the nodes "Cancer" and "Brain Tumor", and "Serum Calcium" and "Coma" were left unoriented since in both cases the two nodes are found to be dependent, and additionally both are in each other's lists of possible parents.

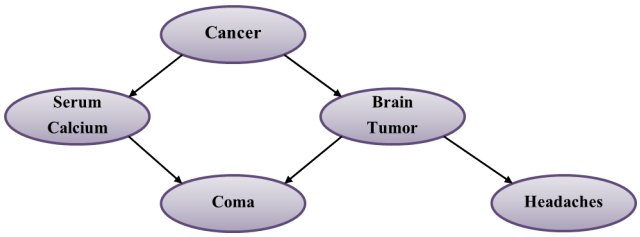


Figure 5.8: The resulting directed acyclic graph learned with PC-TL for the target temporal nodes Bayesian network.

Figure 5.8 shows the DAG obtained after applying the second phase of PC-TL to orient the remaining undirected edges.

5.1.3 Refining the Intervals with Transfer Learning

In this step the information from the structure is used to refine the intervals for each TN. Temporal nodes are addressed in a top down fashion according to their position in the structure. The process begins by obtaining the possible configurations for the parents of a TN, where each configuration is used to generate continuous data from which new intervals are obtained. The idea behind the inclusion of the structural information is that by knowing the parents of each TN, it is possible to leverage this knowledge to obtain intervals tailored to each configuration.

Just as for obtaining the initial intervals, transfer learning is also applied in this step. First, for each auxiliary source domain containing the TN, the possible parental configurations are obtained; the idea being that knowledge transfer will occur between equal configurations of the target domain and the auxiliary domains. However, because auxiliary domains can differ in the structure of the model, their configurations may not be the same as those obtained for the target model. Four scenarios are identified:

1. The target TN and the auxiliary TN have the same parents. In this case the configurations are equal, and no additional operations need to be done.

2. The auxiliary TN has more parents than the target TN. For this scenario the additional parents are marginalized over to obtain the required substructure.

3. The auxiliary TN has fewer parents than the target TN. In this case the auxiliary configuration is repeated for each additional parent in the target network. For example, if the TN has as parents (X, Y, Z) in the target structure, and (X, Y) in the auxiliary structure, then the configurations for (X, Y) are repeated for all values of Z.

4. A combination of scenarios 2 and 3. In this case, first the additional parents are marginalized over, and then the process described in case 3 is applied.

With the auxiliary configurations in the desired form, knowledge transfer can now be applied. For each configuration, continuous data is generated from the intervals belonging to that TN, where each interval generates data in the proportion described by the configuration. Note that each configuration corresponds to a column in the conditional probability table for the TN (assuming the standard approach for representing conditional probability tables). This process is done to generate data from both the target domain and the auxiliary domains. The procedure for determining the total amount of data generated (target + auxiliary domains), and for determining how much data each auxiliary domain generates is the same as for the initial interval approximation method described in Section 5.1.1. The result of this process is a data set for each configuration of the TN in the target domain.

It is important to mention that for data generation to take place, the parameters for the TN of interest given its parents must be calculated. While this is not an issue for the auxiliary sources as these are already learned models, the conditional probability tables for the target domain must be learned previous to the data generation. Two approaches can be taken to determine the TN's conditional probability distribution. The simplest approach involves using only the target data for estimating the conditional probability table; however, the smaller the size of the data set, the worse the estimation will be, leading to an improper representation of the target domain in the total amount of generated data. This ultimately

has a negative impact in the refined intervals. The second approach uses transfer learning to obtain a more reliable estimation of the conditional probability distribution for the TN. This approach is discussed in further detail in Section 5.1.4.

For each obtained data set, the Gaussian mixture model is applied to obtain a set of clusters corresponding to the intervals. Just as in the LIPS algorithm described in Section 3.2.2, the newly obtained interval sets are combined with those obtained from the rest of the data sets, resulting in different possible interval sets.

Finally, of these possible sets, the best is selected using cluster analysis; the goal being to choose clusters with a high intra-cluster density and that are also well separated from other clusters. The metric used for evaluating clusters is the Dunn Index; this metric is discussed in further detail below. It is important to mention that only interval sets with more than one interval and less than five intervals are considered. This reduces evaluations, and keeps the network complexity low.

The Dunn Index

The Dunn index (DI) [Dun73] is a metric used for evaluating clustering algorithms. The objective of the metric is to identify clusters that are compact and well separated from other clusters. In other words, it favors clusters with a low intra-cluster variance and a high inter-cluster distance. The Dunn index is defined to be the ratio between the minimal inter-cluster distance to the maximal intra-cluster distance; it is calculated with the following formula:

$$DI = \min_{1 \leq i \leq n} \left\{ \min_{1 \leq j \leq n, i \neq j} \left\{ \frac{d(i,j)}{\max_{1 \leq k \leq n} d'(k)} \right\} \right\} \tag{5.1}$$

where n is the number of clusters, $d(i,j)$ is the distance between cluster i and j, and $d'(k)$ is the intra-cluster measure of cluster k. A higher Dunn index indicates better clustering, and thus the set of clusters that returns the highest score is determined to be the best set.

The measures $d(i,j)$ and $d'(k)$ may be calculated in a variety of ways, for example, intra-

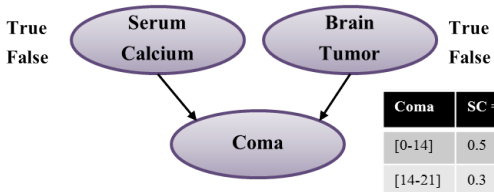
cluster distance could be defined as the distance between the farthest two points within the cluster, or as the average distance between all points to the centroid. For this thesis $d(i, j)$ is defined to be the mean of all pairwise distances between the data points in cluster i to the data points in cluster j; and $d'(k)$ is established to be the mean of all pairwise distances between data points within cluster k. Distances are calculated using the Euclidean distance formula.

A limitation of the Dunn index is its high computational cost as the number of clusters and the number of the data samples increases. While the number of clusters is kept low as to reduce network complexity, it is possible for the amount of generated data to result in a high computational cost. This drawback can be fixed by setting an upper bound on the total amount of data to be generated.

Example of Transfer Learning for Interval Refinement

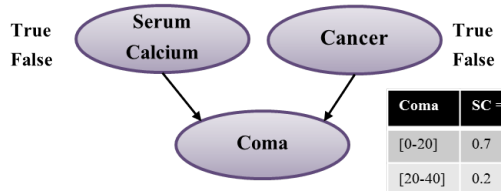
An example of interval refinement with transfer learning is now presented. Consider the TN belonging to the target model shown in Figure 5.9a. The parental configurations for this TN are: $\{SC = true, BT = true\}$ $\{SC = true, BT = false\}$ $\{SC = false, BT = true\}$ and $\{SC = false, BT = false\}$. Now consider the same TN for an auxiliary mode shown in Figure 5.9b. In order to transfer data between the auxiliary and target domain, the structures must first be transformed to have equal configurations. In this case, the auxiliary model has an additional parent (Cancer) but is also missing a parent (Brain Tumor). Therefore, the additional parent must first be removed by marginalizing over it, and then the resulting configuration repeated for the missing parent. Figure 5.10 shows the result of marginalizing over "Cancer" for the auxiliary TN.

Now that the configurations for the auxiliary model have the proper form, data can be generated for each configuration of the target domain. Consider the configuration $\{SC = true, BT = true\}$. If 500 records are to be generated, and the target domain accounts for 25% of these records, that is 125 records, then the remaining 375 records will be



Coma	SC = T, BT = T	SC = T, BT = F	SC = F, BT = T	SC = F, BT = F
[0-14]	0.5	0.4	0.5	0.1
[14-21]	0.3	0.2	0.3	0.05
[21-42]	0.1	0.2	0.1	0.05
Default	0.2	0.2	0.2	0.8

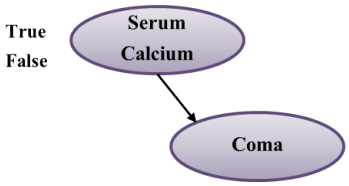
(a) The temporal node "Coma" for the target domain with parents "Serum Calcium" and "Brain Tumor". The conditional probability table is also shown.



Coma	SC = T, Ca = T	SC = T, Ca = F	SC = F, Ca = T	SC = F, Ca = F
[0-20]	0.7	0.5	0.6	0.08
[20-40]	0.2	0.2	0.1	0.02
Default	0.1	0.3	0.3	0.9

(b) The temporal node "Coma" for an auxiliary domain with parents "Serum Calcium" and "Cancer". The conditional probability table is also shown.

Figure 5.9: An example of a temporal node where the parents differ between target and auxiliary domains.



Coma	SC = T	SC = F
[0-20]	0.6	0.4
[20-40]	0.15	0.2
Default	0.25	0.4

Figure 5.10: The result of marginalizing over "Cancer" in the auxiliary model.

contributed by the auxiliary source. For configuration $\{SC = true, BT = true\}$ of the target domain, the configuration $\{SC = true\}$ from the auxiliary domain is used. Note that the configuration $\{SC = true\}$ would also be used for the configuration $\{SC = true, BT = false\}$ since it is repeated for all values of the missing parent.

Since data is generated in the proportion described by the configuration, of the 125 records from the target domain, 50% will be data generated from the interval "$[0 - 14]$", 30% from "$[14 - 21]$", and 10% from the interval "$[21 - 42]$". No records are generated for the "Default" value since it indicates that the temporal event did not occur. For the auxiliary source, of the 375 records 60% will come from the data generated from the interval "$[0 - 20]$" and 15% from the interval "$[20 - 40]$". The combination of the data generated for the target domain and the data generated for the auxiliary domain forms the data set for the configuration $\{SC = true, BT = true\}$. This process is then repeated for all remaining configurations. Lastly, the Gaussian mixture model is applied to each data set to obtain different sets of intervals that are subsequently combined as shown in the example for the LIPS algorithm in Section 3.2.2. Of the resulting possible sets, the best is selected with the Dunn index using the entire collection of generated data to calculate the metric.

5.1.4 Learning the Parameters with Transfer Learning

The final step needed to fully learn a TNBN is to calculate the parameters for each node in the model. A different strategy is taken in order to estimate the probability tables depending on the type of node (instantaneous or temporal).

Parameter Learning for Instantaneous Nodes

For instantaneous, the approach proposed in [LSM10] and discussed in 4.4.1, is used to combine the conditional probability tables for the target task and the auxiliary tasks using weighted mean. If the instantaneous node in the auxiliary structure does not have

the same parents as its counterpart in the target structure, then the appropriate alterations to the auxiliary structure is made. Subsequently, the probability tables are combined with a weighted mean (Equation 4.7). The weight w_i given to each probability table is a value between 0 and 1, where the sum over all weights is equal to 1. The value of w_i is determined by the size of the data set for that domain, where a higher value is given to domains with data sets of greater size with respect to the dimensionality. The larger the size of the data set, the more reliable it is considered to be.

Parameter Learning for Temporal Nodes

Because a temporal node in the target domain may not have the same values (i.e., intervals) as its counterpart in an auxiliary domain, the strategy used for instantaneous nodes cannot be directly applied to TNs. Two transfer learning strategies are used for calculating the conditional probability tables for TNs. The first approach is only used when records are generated for refining the intervals as discussed in the previous section. This transfer learning strategy uses an interval mapping strategy to give a good approximation of the conditional probability distribution, so that the target domain is more properly represented in the data that is generated. The second approach estimates the conditional probability distributions with maximum likelihood.

Temporal Node Parameter Learning with Interval Mapping

The problem in applying the strategy used for instantaneous nodes to TNs is the absence of value equality for the interval sets of two domains. However, if the data sets (with temporal information in discrete form) for the auxiliary sources are available, a mapping procedure can take place to transform the conditional probability tables for the auxiliary domains to use the same interval values as the target domain. By assuming a Gaussian distribution to characterize each interval, a continuous value can be generated and then mapped to the

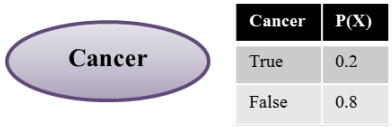
correct interval for the target domain. If this procedure is applied to the temporal information for each record of the auxiliary data set, the result will be a transformed data set that uses the same intervals as the target domain. The conditional probability tables can then be recalculated and combined in the same manner as for instantaneous nodes. Note that if the data sets for the auxiliary sources contain the original continuous data, these values can be directly mapped to the appropriate interval, and it is no longer necessary to generate Gaussian values. Additionally, if the temporal ranges for the target and auxiliary domains do not intersect, that is they are completely different, mapping cannot take place and the conditional probability tables will be calculated using only the scarce data from the target domain.

Temporal Node Parameter Learning with Maximum Likelihood

The procedure described above is used to give a good estimation of the conditional probability distribution for a TN with the unrefined intervals, that is, the initial approximation. Once the intervals have been refined, the parameters for the TN will have to be recalculated for the new values. If the procedure described in Section 5.1.3 is recalled, the result of applying transfer learning is a data set for each configuration of the TN's parents. Since each configuration corresponds to a column of the desired conditional probability table, it is only necessary to transform the generated data sets to use the found intervals instead of the continuous data by replacing each continuous value with the interval it falls in. Once this transformation has taken place, the parameters are estimated with maximum likelihood.

Example of Transfer Learning for Parametric Learning

An example of transfer learning for parameter learning is now provided. Consider the instantaneous node "Cancer". This node is shown is Figures 5.11a and 5.11b for the target domain and an auxiliary domain respectively. In order to apply knowledge transfer to esti-



Cancer	P(X)
True	0.2
False	0.8

(a) The instantaneous node "Cancer" for the target domain with its probability table.

Cancer	P(X)
True	0.4
False	0.6

(b) The instantaneous node "Cancer" for an auxiliary domain with its probability table.

Cancer	P(X)
True	0.36
False	0.64

(c) The transferred parameters for the instantaneous node "Cancer", resulting from the combination of the probability tables from the target domain and an auxiliary domain.

Figure 5.11: An instantaneous node for the target domain and an auxiliary domain with their corresponding probability distributions. Figure 5.11c shows the result of learning the parameters with transfer learning.

mate the probability table for the target model, it is first necessary to calculate the weight associated to each domain. Consider an auxiliary domain with 500 records and a target domain with 125 records for a total of 625 records. Of the total, the auxiliary domain accounts for 80% and the target domain for 20%. Equation 4.7 can now be used to combine the probability tables as follows:

$$\begin{aligned} P\left(Cancer = true\right) &= 80\% \times P_{aux}(true) + 20\% \times P_{target}(true) \\ &= 0.8 \times 0.4 + 0.2 \times 0.2 = 0.36 \\ P\left(Cancer = false\right) &= 80\% \times P_{aux}(false) + 20\% \times P_{target}(false) \\ &= 0.8 \times 0.6 + 0.2 \times 0.8 = 0.64 \end{aligned}$$

Figure 5.11c shows the resulting probability distribution for the instantaneous node "Cancer" of the target domain.

Now consider the TN "Coma". When refining the intervals, a data set for each parental

$SC = T, BT = T$	$SC = T, BT = F$	$SC = F, BT = T$	$SC = F, BT = F$
1	5	39	17
15	42	26	33
7	23	13	24
29	16	2	15
40	14	8	21
38	11	18	5
17	38	34	13
12	4	9	35
31	25	12	27
8	16	40	7

Table 5.3: An example of the data sets created for each parental configuration of the temporal node "Coma". For simplicity only 10 records are shown.

$SC = T, BT = T$	$SC = T, BT = F$	$SC = F, BT = T$	$SC = F, BT = F$
[0-10]	[0-10]	[32-42]	[10-32]
[10-32]	[32-42]	[10-32]	[32-42]
[0-10]	[10-32]	[10-32]	[10-32]
[10-32]	[10-32]	[0-10]	[10-32]
[32-42]	[10-32]	[0-10]	[10-32]
[32-42]	[10-32]	[10-32]	[0-10]
[10-32]	[32-42]	[32-42]	[10-32]
[10-32]	[0-10]	[0-10]	[32-42]
[10-32]	[10-32]	[10-32]	[10-32]
[0-10]	[10-32]	[32-42]	[0-10]

Table 5.4: The results of replacing the continuous records in the data sets of Table 5.3 with the selected intervals.

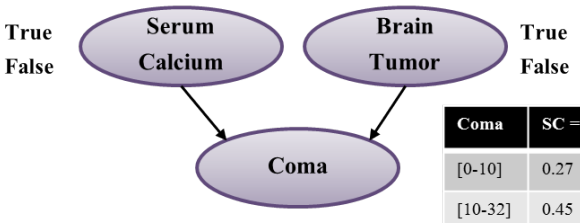
configuration was created. Table 5.3 shows an example of the data sets. For simplicity, in this example the data sets consist only of 10 records.

Assume the interval set $\{[0-10], [10-32], [32-42]\}$ was selected with the Dunn index in the previous step. Each continuous record of the data sets can now be replaced with the interval that it falls in. Table 5.4 shows the result of this procedure.

The parameters for the temporal node are now calculated with maximum likelihood, where the result for each configuration corresponds to a column in the conditional proba-

Coma	$P(SC = T, BT = T)$
[0-10]	0.27
[10-32]	0.45
[32-42]	0.18
Default	0.1

Table 5.5: The probability distribution for "Coma" given "Serum Calcium" and "Brain Tumor" are both true.



Coma	SC = T, BT = T	SC = T, BT = F	SC = F, BT = T	SC = F, BT = F
[0-10]	0.27	0.16	0.24	0.06
[10-32]	0.45	0.48	0.32	0.18
[32-42]	0.18	0.16	0.24	0.06
Default	0.1	0.2	0.2	0.7

Figure 5.12: The transferred parameters for the temporal node "Coma", resulting from applying maximum likelihood to the data sets created for each parental configuration in the interval refinement step.

bility table. For example, for configuration $\{SC = true, BT = true\}$, 30% of the records are "$[0 - 10]$", 50% are "$[10 - 32]$" and 20% are "$[32 - 42]$". These values are subsequently normalized to consider the percentage of records of the configuration that are "Default". The probability distribution for configuration $\{SC = true, BT = true\}$ with 10% of "Default" records is shown in Table 5.5.

The same procedure is applied to the remaining configurations. Figure 5.12 shows the resulting conditional probability distribution for the TN "Coma".

5.2 Chapter Summary

In this chapter the methodology for learning a Temporal Nodes Bayesian Network with transfer learning was presented. A Transfer learning strategy was applied to each of the

necessary steps for learning a TNBN, and an example for each step was presented in order to provide more clarity to how knowledge transfer is being used. This algorithm constitutes the main contribution of this thesis. Unlike traditional machine learning algorithms (e.g. LIPS) which assume that sufficient data is available for training, the TNBN-TL algorithm can learn reliable models even when data is scarce. In the following chapters, the experiments carried out to test the methodology are presented. Experiments were done with synthetic data modeling the event of an automobile accident, and real world data that models the mutational networks for the Human Immunodeficiency Virus.

Chapter 6

Experiments with Synthetic Data

In this chapter a series of experiments done to evaluate the transfer learning algorithm proposed in this thesis are presented. The experiments are performed using synthetic data that models the event of a collision. Section 6.1 presents the metrics used to evaluate the learned models. Subsequently, the data used for the experiments is described; and the last section presents the experiments carried out, along with an analysis of the results.

6.1 Evaluation Metrics

To judge the resulting models, the following elements were evaluated:

- The predictive precision of the model
- The accuracy in estimating the temporal events
- The accuracy of the induced structure when compared with the true structure

The predictive precision of a learned model is evaluated with the Relative Brier Score (RBS) [Bri50]. The RBS gives an evaluation of the entire model by instantiating a random set of nodes, and then inferring the remaining hidden nodes. Expressed as a percentage, the RBS is defined as:

$$RBS = \left(1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{r}(E_{ij} - P_{ij})^2\right) \times 100 \tag{6.1}$$

where n is the sample size, r is the number of values the node can take, E_{ij} is a value of 1 or 0 depending on whether the event occurred in value j of the node or not, and P_{ij} is the marginal posterior probability of the value j for each node given the evidence. Note that a score of 100% indicates a perfect prediction.

For evaluating the accuracy of the learned model in estimating temporal events, the following metric was defined:

$$Time\,Difference = \frac{t_e - e}{Interval_{range}} \tag{6.2}$$

where t_e is the middle point of the interval where the event e occurs, and $Interval_{range}$ is its range. This metric does not consider prediction, since the interval where an event occurs is not necessarily the predicted interval. Smaller values are desirable for this metric since the aim is to minimize the distance from the actual time the events occur, to the middle points of the interval where they occur.

The quality of the induced structures was assessed by calculating the edit distance with the target model. Note that in order to calculate this metric it is necessary to have the *true* TNBN structure for using as a *gold standard*. The edit distance is calculated by counting the number of missing and added arcs (including inverted arcs). A lower value indicates a structure closer to the *true* structure, with zero being a perfect score.

6.2 Data Sets

For these experiments, the TNBN model presented in Figure 6.1 is used as the target model to be learnt. This TNBN presents the possible consequences of a collision as modeled in [HMG95]. For this model the nodes "Collision" (C), "Head Injury" (HI) and "Internal Bleeding" (IB) are all instantaneous nodes. Because the symptoms of a head injury or internal

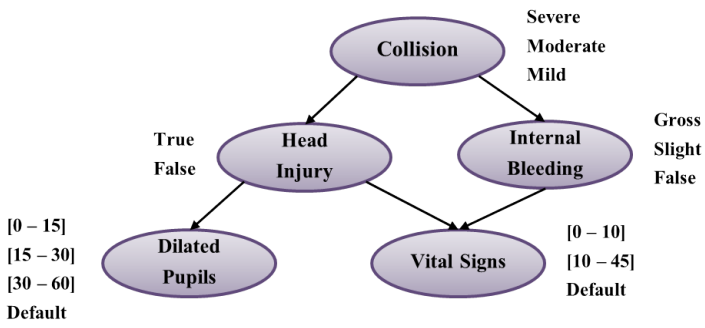


Figure 6.1: The target Temporal Nodes Bayesian Network that models the event of a collision. "Collision", "Head Injury" and "Internal Bleeding" are instantaneous nodes, and "Dilated Pupils" and "Vital Signs" are temporal nodes each with a set of intervals and a default value.

bleeding may not be present immediately, "Dilated Pupils" (DP) and "Vital Signs" (VS) are modeled by temporal nodes. For example, the vital signs of a person who has been in a collision may become unstable within 0 to 10 minutes, 10 to 45 minutes, or not at all (Default), if a head injury or internal bleeding occurs.

A data set of 2200 records for this model was generated using the Elvira System [C+02]; however, because this model has already been learned, all the components of the TNBN are fully defined and the generated data for the temporal nodes is in the form of discrete temporal intervals. Consequently, it is necessary to transform the temporal intervals of the data set to continuous form. In order to do this, it is assumed that each temporal interval is described by a Gaussian distribution, where the midpoint of interval is the mean, and the distance of the midpoint to either extreme the standard deviation. With this assumption, a random numerical value belonging to the range $\mu \pm \sigma$ is generated to replace the temporal interval. This process is repeated for each interval in the data resulting in a data set where the temporal information is continuous. After applying this transformation, the data set was subsequently divided into a training set with scarce data samples, and a testing set.

6.2.1 The Auxiliary Source Domains

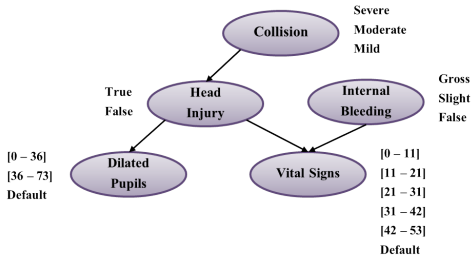
Auxiliary source models were created by using the TNBN model of Figure 6.1 as a base model, and subsequently introducing various alterations to the networks. Changes were introduced to each component of the TNBN as follows:

- The structures of the models were changed by randomly adding or removing links between nodes, while making sure no cycles were formed.
- Gaussian noise with a mean of 1 and different standard deviations was introduced into the conditional probability tables for each node of the created network.
- The temporal nodes for the models were altered in the following ways. First the entire temporal range was changed by subtracting or adding a random percentage of the range. For example, for the node "Vital Signs" in Figure 6.1, the entire temporal range is 45, if 20% of the range were added, the new range would be 54 (i.e., $45 + (0.2) \times 45$). This new range is then divided between a randomly selected number of intervals, and the probability for all intervals (that is, $1 - P(Default)$) is uniformly divided between them.

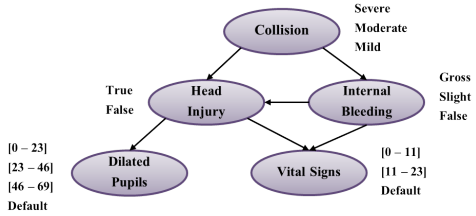
For these experiments, three auxiliary models were created where each model holds a different degree of similarity to the target model. Table 6.1 describes the alterations done for each of the three models. All three auxiliary TNBNs can be seen in Figure 6.2. Just as for the target model, records for each auxiliary network were generated using the Elvira System, where $Aux1$, $Aux2$ and $Aux3$ have 540, 560 and 480 records, respectively.

Model	Structure	Parameters	Temporal Nodes	
			% of range + or −	# of intervals
$Aux1$	1 link removed	$\sigma = 0.5$	DP: 22% added VS: 17% added	DP: 2 VS: 5
$Aux2$	1 link added	$\sigma = 1$	DP: 15% added VS: 48% subtracted	DP: 3 VS: 2
$Aux3$	1 link added 1 link removed	$\sigma = 3$	DP: 12% subtracted VS 37% subtracted	DP: 2 VS: 2

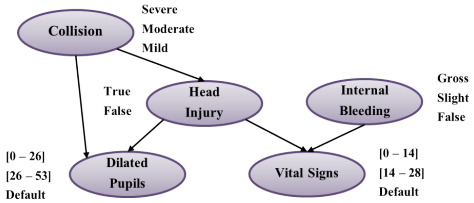
Table 6.1: A description of the alterations made for each auxiliary source model. The column "Structure" indicates the links added or removed, while "Parameters" shows the standard deviation used to generated Gaussian noise. For the temporal nodes, the percentage of the original temporal range added or subtracted to "Dilated Pupils" (DP) and "Vital Signs" (VS) is provided, along with the number of intervals this new temporal range was divided into.



(a) $Aux1$



(b) $Aux2$



(c) $Aux3$

Figure 6.2: Three auxiliary source models created by altering the structure, the parameters and the temporal information of the target model of Figure 6.1.

6.3 Experiments

In the following experiments the proposed algorithm is evaluated by attempting to learn the target model presented in Figure 6.1 from data sets with scarce records and the three auxiliary domains shown in Figure 6.2. The data set consisting of 2200 records that was generated for the target model was partitioned into training and test sets. To evaluate the algorithm, three different sizes of training sets were used to learn models (10, 44 and 100 records). Each experiment was repeated ten times to gain a better estimate of how the learned models behave, and for each repetition a new training set was used for learning. It is important to mention that no data samples were shared among the ten training sets, that is, these sets were disjoint.

When learning the structure, a node ordering was provided. This particular domain allows certain assumptions on the possible orderings to be made, since it is quite obvious that the "Collision" is the catalyst for all other events, and not the other way around. The following order was provided:

$Collision \leftarrow \{\}$
$Head\,Injury \leftarrow \{Collision\}$
$Internal\,Bleeding \leftarrow \{Collision\}$
$Dilated\,Pupils \leftarrow \{Collision, Head\,Injury, Internal\,Bleeding\}$
$Vital\,Signs \leftarrow \{Collision, Head\,Injury, Internal\,Bleeding\}$

Results of the experiments are compared using a T-test to determine statistical significance at a 95% confidence level.

6.3.1 Evaluating the Effects of Using a Partial Order for Structural Learning

The objective of this experiment is to evaluate how the inclusion of a partial order affects the learned models, the hypothesis being that it will improve the resulting structures. For this

Data Set Size	Metric	No Order	Partial Order
10	RBS	70.11 (4.79)	**70.29 (5.42)**
	Time Difference	**0.27 (0.01)**	**0.27 (0.01)**
	Edit Distance	2.5 (0.81)	**2.3 (0.9)**
44	RBS	76.12 (1.12)	**77.07 (0.72)** †
	Time Difference	0.28 (0.005)	**0.28 (0.004)**
	Edit Distance	2.1 (0.7)	**1.4 (0.49)** †
100	RBS	77.13 (1.6)	**77.25 (1.43)**
	Time Difference	0.28 (0.005)	**0.28 (0.004)**
	Edit Distance	3.2 (0.6)	**1.6 (0.49)** †

Table 6.2: Comparison of the results of learning a TNBN with the TNBN-TL algorithm using a partial order vs. no partial order. The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol denotes that a statistically significant difference exists between two values.

experiment models are learned using the proposed TNBN-TL algorithm, and the results of including the partial order previously described are compared to the results obtained when no partial order is provided. Table 6.2 shows the comparison of the two outcomes using different data set sizes. A '†' symbol is used to express a significant difference in the results.

Results

From the results displayed in Table 6.2 the following observations are made:

- In all cases, the measured edit distances for models learned using a partial order is superior to the edit distances obtained by models learned without a partial order.
- Models learned using a partial order obtained a better RBS than those learned without a partial order.

- The inclusion of a partial order does not appear to affect the measured time difference. For both types of models, the measured time difference is maintained low and remains stable for the different data set sizes.

By including a node ordering, edit distances are reduced in all cases. The difference between the edit distances measured for models learned with a partial order and models learned without a partial order becomes more prominent as the sizes of the data sets increase. It should be noted that with the smallest data set (10 records), the improvement obtained by including a partial order is not considered significant at the 95% confidence level. However, as more records become available the inclusion of a partial order has a higher impact and thus, becomes more significant.

Interestingly, when learning with the largest data set, both models show an increase in their measured edit distances (this could be a result of confusion brought on by a larger target data set, leading to incorrect leveraging of knowledge by PC-TL). However, by including the partial order this negative impact is maintained low, while models learned without the partial order suffer a more dramatic decrease in structure accuracy.

Besides obtaining a more accurate structure, the inclusion of a partial order allows for saving on computational resources by reducing the amount of independence tests performed in PC-TL. On average, models learned without a partial order calculated 55 independence tests. This number was reduced to 44.3 for models learned with a partial order, for an average savings of 10.7 calculations. While this particular domain is small, in a domain that models a larger number of variables, the benefits of avoiding such tests would become more visible, since the temporal complexity of PC-TL is dependent on the number of nodes in the network.

6.3.2 Evaluating the Effects of Refining the Intervals

In this set of experiments, models learned with interval refinement are compared to models that only use the initial approximation, i.e., the k-means algorithm. The purpose of this

Data Set Size	Metric	Without Refining Intervals (Avg.)	Refining Intervals
10	RBS	**72.56 (2.22)**	70.29 (5.42)
	Time Difference	**0.24 (0.01)** †	0.27 (0.01)
	Edit Distance	**2.13 (0.67)**	2.3 (0.9)
44	RBS	73.99 (1.02)	**77.07 (0.72)** †
	Time Difference	**0.24 (0.01)** †	0.28 (0.004)
	Edit Distance	1.7 (0.37)	**1.4 (0.49)**
100	RBS	74.66 (1.09)	**77.25 (1.43)** †
	Time Difference	**0.24 (0.01)** †	0.28 (0.004)
	Edit Distance	1.8 (0.14)	**1.6 (0.49)**

Table 6.3: Comparison of the results of learning a TNBN without interval refinement vs. with interval refinement. The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol denotes that a statistically significant difference exists between two values.

experiment is to evaluate if the structural information is improving the intervals obtained for a TN. Because k-means requires the number of clusters as input, the performance of the models built without refinement will depend directly on the selected value. All possible combination of values were given to k-means as input, from a minimum of 2 to a maximum of 4 intervals, for each TN. An average of the results was calculated and compared to the results of learning the models with refinement. Table 6.3 shows the outcome of this experiment.

Results

From the results displayed in Table 6.3 the following observations are made:

- Models learned without interval refinement are obtaining a better measured time difference.

- As more target records are used, the measured RBS for models learned with interval refinement improves, and is significantly better than the RBS displayed by models learned without interval refinement.

- No significant improvement is obtained for the edit distances; however, this is expected since both types of models are learned using a partial order.

Initially, it appears that refining the intervals is hindering the resulting models rather than improving them, since for a target data set of 10 records all metrics are reported to be superior when no interval refinement is taking place. However, neither the improvement in the RBS nor the edit distance is significant at the 95% confidence level.

While with all data set sizes, the measured time difference for models learned without refinement, does obtain a statistically significant improvement over the reported time differences for models learned with interval refinement, it is still necessary to weigh this improvement against the behavior displayed by the RBS. For data sets of sizes 44 and 100, the RBS obtained by models learned with interval refinement is significantly better than the reported RBS for models without refinement.

It is important to note that a TNBN model displays a compromise between predictive precision and temporal accuracy. When the temporal range of a TN is divided over a small amount of intervals, the number of events that can fall within the range of an interval increases. This results in higher probabilities, and as a consequence, better predictive precision. However, the larger interval ranges also result in events occurring farther from the expected time, that is, the midpoint of the interval. Consequently, intervals with larger ranges have a greater measured time difference, meaning they provide a less accurate estimation of the point in time when an event occurs. In contrast, a temporal range divided over a greater number of intervals reduces the number of events that can fall within the range of an interval, thus lowering probabilities and decreasing the predictive precision. However, because the intervals are smaller, the measured time difference is also reduced, resulting in better estimations about the point in time when an event occurs.

Models learned without interval refinement are obtaining a better measured time difference but a worse RBS, meaning they are favoring temporal accuracy. In contrast, interval refinement results in a worse measured time difference but a superior predictive precision. It is important to mention that learning without interval refinement holds one important disadvantage to learning with interval refinement. It requires knowledge about the number of intervals each TN has, and this information is not always available. By learning with refinement the need for guessing when selecting intervals for each TN is eliminated, and the risks of obtaining poor models as a result of selecting an improper amount of intervals are avoided.

6.3.3 TNBN-TL vs. Naive Transfer

In the following experiment the results of learning with the proposed TNBN-TL algorithm are compared to the results of learning by performing naive transfer. In naive transfer, the data sets for the auxiliary source domains are simply combined with the target data set, and a model is learned from this new augmented data set using a traditional machine learning algorithm. No consideration is given to the similarities between auxiliary sources and the target source, and so, unlike the TNBN-TL algorithm, knowledge is not leveraged based on strength of relationships, rather transfer is carried out indiscriminately.

Table 6.4 shows the comparison of the results obtained with the TNBN-TL algorithm and naive transfer.

Results

From the results displayed in Table 6.4 the following observations are made:

- Models learned with the TNBN-TL algorithm show a dramatically superior predictive precision than models learned using naive transfer.

Data Set Size	Metric	Naive Transfer	TNBN-TL
10	RBS	55.08 (0.38)	**70.29 (5.42)** †
	Time Difference	0.35 (5.38×10^{-04})	**0.27 (0.01)** †
	Edit Distance	**1.0 (0)** †	2.3 (0.9)
44	RBS	55.29 (1.54)	**77.07 (0.72)** †
	Time Difference	0.36 (0.02)	**0.28 (0.004)** †
	Edit Distance	**1.0 (0)** †	1.4 (0.49)
100	RBS	56.21 (2.03)	**77.25 (1.43)** †
	Time Difference	0.35 (0.01)	**0.28 (0.004)** †
	Edit Distance	**1.0 (0)** †	1.6 (0.49)

Table 6.4: Comparison of the results of learning a TNBN by performing naive transfer (i.e., combining the data sets from all sources) and learning with the TNBN-TL algorithm. The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol denotes that a statistically significant difference exists between two values.

- The measured time differences for models learned with the TNBN-TL algorithm are significantly better than the measured time differences reported for models learned with naive transfer.
- Better structural accuracy is obtained when learning models with naive transfer.

The goal of this experiment is to evaluate how well the proposed TNBN-TL algorithm is leveraging the similarities between the auxiliary domains and the target domain. Naive transfer treats disparate auxiliary domains the same way it treats similar auxiliary domains, so negative transfer is more likely to occur since it makes no effort to minimize the impact of those elements that hold a weaker relationship to the target domain. In addition, transferring in this manner allows the auxiliary information to overwhelm the target data resulting in less reliable models, though it should be noted that if the auxiliary domains are very similar to the target domain, results may be comparable.

From the reported results, it is observed that when learning with naive transfer the target data becomes overshadowed by the auxiliary data. This results in poorer predictive precision and temporal accuracy. In contrast, the TNBN-TL algorithm is able to control *how* transfer occurs and *what* is transferred, obtaining superior results for both metrics.

The edit distances reported for models learned with naive transfer is lower compared to the reported values for the TNBN-TL algorithm. Note that for this synthetic domain, structures for the auxiliary models were obtained by using the target model as a base. Therefore, when combining data sets, the relationships between nodes that are common to more domains are reinforced, while the impacts of those particular to a single domain are reduced. This advantage becomes prominent as more auxiliary domains are made available, since there is more data to reinforce positive transfer and reduce negative transfer. In addition, this particular benefit of naive transfer is not seen with the PC-TL algorithm, since transfer occurs only between the target domain and the most similar auxiliary domain instead of all auxiliary domains. It should be noted that while this behavior holds for this particular domain, it would not necessarily be true for other real-world domains where many shared dependencies and independencies between auxiliary and target domains cannot be guaranteed.

6.3.4 Evaluating the Effects of Learning with Transfer

The following experiments measure the effects of using transfer learning to induce a TNBN when scarce data is available for training. The results obtained with the TNBN-TL algorithm are compared with the results obtained with two different algorithms that do not use transfer, that is, they learn solely from the scarce data sets available for the target domain.

To learn without transfer, a version of the algorithm proposed in this thesis that does not use knowledge transfer was implemented. This algorithm, referred to as TNBN-PC, uses the original PC algorithm [SGS00], modified to include a node ordering, to learn the structure. Intervals for the model are learned using only the continuous data from the target domain, and parameters are induced with maximum likelihood.

Data Set Size	Metric	LIPS	TNBN-PC	TNBN-TL
10	RBS	46.29 (9.25)	45.78 (8.31)	**70.29 (5.42)** †∗
	Time Difference	0.49 (0.09)	0.49 (0.09)	**0.27 (0.01)** †∗
	Edit Distance	2.5 (1.2)	3.6 (0.49)	**2.3 (0.9)** ∗
44	RBS	64.71 (6.79)	58.59 (7.96)	**77.07 (0.72)** †∗
	Time Difference	0.36 (0.08)	0.39 (0.07)	**0.28 (0.004)** †∗
	Edit Distance	**0.6 (0.92)** †	1.7 (0.9)	1.4 (0.49)
100	RBS	66.41 (5.34)	59.07 (5.1)	**77.25 (1.43)** †∗
	Time Difference	0.34 (0.04)	0.38 (0.06)	**0.28 (0.004)** †∗
	Edit Distance	**0.5 (0.67)** †	2.9 (0.83)	1.6 (0.49) ∗

Table 6.5: Comparison of the results of learning the TNBN with the TNBN-TL algorithm (with transfer) and learning the model with the TNBN-PC algorithm and the LIPS algorithm (without transfer). The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol denotes that a statistically significant difference exists between the values measured for the TNBN-TL algorithm and the LIPS algorithm, while a '∗' symbol expresses a significant difference between the TNBN-TL algorithm and the TNBN-PC algorithm.

In addition, the LIPS algorithm 3.2.2 was also employed to learn models solely from the target data. Since the LIPS algorithm uses the K2 algorithm to learn the structure, a node ordering is required. The correct node ordering (Collision, Head Injury, Internal Bleeding, Dilated Pupils, Vital Signs) for the domain was provided for the K2 algorithm, while the TNBN-PC algorithm and the TNBN-TL algorithm used the partial order previously defined.

Table 6.5 shows the comparison of the results of learning the TNBN with transfer learning using the TNBN-TL algorithm, and learning the TNBN with the TNBN-PC algorithm and the LIPS algorithm. A '†' symbol is used to express a significant difference in the results between the LIPS algorithm and the TNBN-TL algorithm, while a '∗' symbol expresses a significant difference between the TNBN-PC algorithm and the TNBN-TL algorithm.

Results

From the results displayed in Table 6.5 the following observations are made:

- Models learned with the TNBN-TL algorithm show a dramatically superior predictive precision than models learned with the TNBN-PC algorithm and the LIPS algorithm, both of which do not use transfer learning.
- The measured time differences for models learned with the TNBN-TL algorithm are significantly better than the measured time differences reported for models learned without transfer learning.
- The TNBN-TL algorithm obtains more accurate structures compared to the TNBN-PC algorithm. As more target data becomes available, the LIPS algorithm obtains the best measured edit distances.

From the reported results it is observed that transfer learning is indeed improving the resulting models. By leveraging similarities from auxiliary source domains, the TNBN-TL algorithm is able to obtain significant improvements in both the predictive precision and the temporal accuracy of models, achieving in the measured RBS an average improvement of 15.73% over the LIPS algorithm, and an improvement of 20.39% over the TNBN-PC algorithm.

The LIPS algorithm, which employs the K2 algorithm [CH92] to retrieve the structure, achieved better edit distance scores for target data sets of sizes 44 and 100. It is important to be noted, that the LIPS algorithm restricts the number of parents each node can have to 2 in order to keep network complexity low. This makes the collision domain particularly well suited for structure learning with the LIPS algorithm, since in reality no node has more than two parents. In addition, the LIPS algorithm was provided with the correct node ordering, this fact also contributes to the better structural accuracy displayed by the models built with the LIPS algorithm.

6.3.5 Evaluating the Effects of Varying the Degree of Similarity

The last experiments done for the collision domain focus on evaluating how the degree of similarity that exists between an auxiliary source domain and a target domain, impacts the learned models. For this experiment, the three auxiliary domains that were previously created were ranked according to how similar they are to the target domain. To evaluate similarity, the global similarity metric (Equation 4.4) was used, resulting in the following ranking: $Aux1$ (most similar), $Aux2$ (medium similarity) and $Aux3$ (least similar). Models were then learned by transferring from only one domain at a time.

Table 6.6 shows the comparison of the results of learning from varying degrees of similarity. A '†' symbol denotes that a result is significantly better than the result obtained when transferring from the least similar domain, a '‡' that a result is significantly better than the result obtained when transferring from the domain with medium similarity, and a '$*$' symbol denotes that a result is significantly better than the result obtained when transferring from the most similar domain.

Results

From the results displayed in Table 6.6 the following observations are made:

- The RBS reported for models learned by transferring from the most similar domain is significantly better in all cases.
- The measured time differences for models learned by transferring from the most similar domain is comparable for data sets of all sizes. For models learned by transferring from the auxiliary sources of medium and least similarity, better measured time differences are reached when using data set sizes of 44 and 100 records.
- For most cases better structural accuracy is achieved when transferring from the most similar domain.

Data Set Size	Metric	Least	Medium	Most
10	RBS	53.61 (7.55)	51.63 (8.01)	**69.68 (4.79)** †‡
	Time Difference	0.38 (0.09)	0.39 (0.09)	**0.27 (0.01)** †‡
	Edit Distance	**2.5 (0.5)** ‡	4.2 (0.98)	2.6 (0.49) ‡
44	RBS	72.16 (1.14) ‡	70.75 (1.67)	**75.11 (0.57)** †‡
	Time Difference	0.25 (0.01) ∗	**0.24 (0.02)** ∗	0.27 (0.02)
	Edit Distance	**2.0 (0)** ‡	2.9 (0.3)	**2.0 (0)** ‡
100	RBS	73.09 (0.6)	72.84 (1.37)	**76.53 (0.44)** †‡
	Time Difference	0.24 (0.007) ∗	**0.23 (0.009)** †∗	0.28 (0.001)
	Edit Distance	2.2 (0.4)	1.6 (0.49) †	**1.1 (0.3)** †‡

Table 6.6: A comparison of the results of learning from varying degrees of similarity. The three auxiliary domains of Figure 6.2 were ranked from least similar to most similar, and models were learned by transferring from only one domain at a time. The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol denotes that a result is significantly better than the result obtained with the least similar domain, a '‡' that a result is significantly better than the result obtained with the domain with medium similarity, and a '∗' denotes that a result is significantly better than the result obtained with the most similar domain.

The objective of this experiment is to evaluate how different degrees of auxiliary source similarity affect the models learned with the TNBN-TL algorithm. As expected, most metrics obtain better results when transferring from the most similar domain; however, the best measured time difference for data sets of 44 and 100 records are not obtained by the models learned by transferring from the most similar domain. This can be attributed to how similarity is being measured. For this experiment, models are ranked by using the global similarity metric (Equation 4.4), which counts the number of shared dependencies and independencies. This metric does not consider similarity in temporal ranges or temporal intervals, and as a consequence it is possible for the auxiliary domain ranked most similar to hold little resemblance to the temporal values for the target domain. As a result, better measured time

differences are being obtained by auxiliary domains considered to be less similar. Note that this can also contribute to the drop in the measured RBS for the auxiliary domain of medium similarity, as the learned intervals will also contribute to the overall precision of the model. These results indicate a need for defining a similarity metric which considers the temporal component of the TNBN model, and thus would be better suited for evaluating similarity between TNBNs.

Note that even when transferring only from the most similar domain, the reported RBS values are lower than those obtained when transferring from all three auxiliary sources (as reported in the previous experiments for the TNBN-TL algorithm). This proves that the inclusion of more sources, even though they may be less similar, is contributing to obtaining more accurate models.

6.3.6 TNBN-TL Behavior Analysis

Lastly, the behavior of the TNBN-TL algorithm as a function of the number of available target data samples is analyzed. As is expected, the predictive accuracy of a learned model will depend heavily on the amount of records used for training. This remains true even when applying transfer learning to compensate for small training sets. For the synthetic collision domain, the TNBN-TL algorithm showed a logarithmic deterioration rate in terms of the measured RBS (Figure 6.3). As fewer data samples are available for learning, the accuracy of the models begins to descend, at first moderately and then much more rapidly due to the extremely diminished size of the training sample.

It is important to mention that the behavior of the TNBN-TL algorithm and how it degrades as a function of the available target data will differ for each domain as this is dependent on the number of variables and the number of variable values. While the TNBN-TL algorithm behaved in this manner for this particular domain, it may not hold true for other domains where more variables exist in the model.

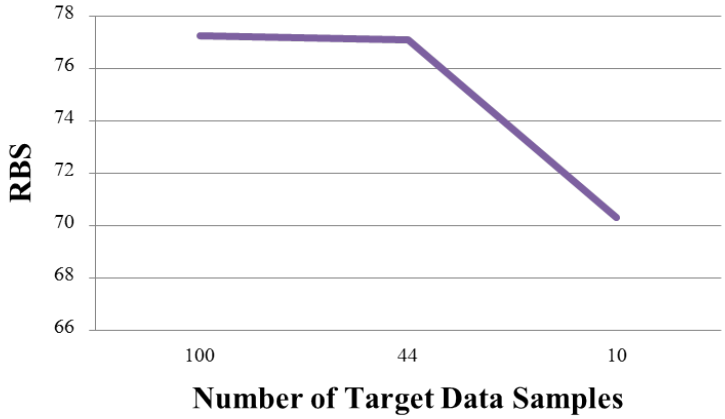


Figure 6.3: The behavior of the predictive accuracy obtained by the TNBN-TL algorithm as a function of the number of available target data samples for the synthetic collision domain.

6.4 Chapter Summary

In this chapter a series of experiments done using synthetic data that models the event of a collision were presented. Experiments were performed to evaluate the benefits of including a node ordering, as well as the benefits of refining the intervals. In addition, the proposed TNBN-TL algorithm was evaluated against the results of learning with naive transfer, and learning without any knowledge transfer. Overall, the models learned with the TNBN-TL performed significantly better than models learned with naive transfer and no knowledge transfer. Superior results were obtained for predictive precision and temporal accuracy by using transfer learning. Experiments were also carried out to evaluate the effects of transferring from varying degrees of similarity. While most metrics obtained better results when transferring from the most similar auxiliary domain, the results of these experiments support a need for defining a new similarity metric better suited for the temporal components of the TNBN model. In the next chapter, experiments done with real-world data are presented.

Chapter 7

Experiments with Real-World Data

In this chapter, experiments done with real-world data are presented. The algorithm proposed in this thesis was applied to the medical domain of the Human Immunodeficiency Virus (HIV) to learn the mutational networks that develop as a response to antiretroviral treatment. Specifically, a mutational network belonging to a region with scarce records (Europe) was learned by transferring knowledge from a region with a larger amount of data (United States). The first section gives an overview of HIV. Subsequently, a brief explanation of the data set used for this experiment is provided. The final section presents the experimental results obtained for this domain.

7.1 The Human Immunodeficiency Virus

The Human Immunodeficiency Virus, or *HIV*, is a disease that effects the human immune system, leaving the host vulnerable to attacks by several opportunistic infections and cancers. If left untreated, HIV can cause *AIDS* (Acquired Immunodeficiency Syndrome) which leads to a progressive failure of the immune system, resulting in severe illness and eventual death.

HIV is considered the primary cause of the current AIDS pandemic. While HIV was first documented in 1983 [BSCR$^+$83], its history dates further back. Scientist believe that HIV jumped from primates to human beings in the early 20^{th} century [SH11]. They believe

the primate version of HIV (Simian Immunodeficiency Virus, or *SIV*) was transmitted to humans when hunters came into contact with infected meat. From there, it spread without being detected until 1981, when the US Centers for Disease Control and Prevention (CDC) published a report about 5 homosexual men experiencing a rare lung infection [fDCP81]. This is considered the first documented case of AIDS. Today AIDS has claimed over more than 25 million lives, and as of 2011 there are approximately 34 million people living with HIV [Org13]. Different subtypes of HIV exist, and can be more prevalent depending on the geographical region.

7.1.1 The Life-cycle of HIV

HIV is transmitted for human to human through sexual contact, blood (e.g. transfusions with infected blood, or contaminated needles), or through breast milk from mother to child. Once a person has been infected with HIV the virus begins to spread by attacking various cells. The primary targets of HIV are CD4 lymphocyte cells, or CD4 cells. These cells are a crucial part of the human immune system, and by compromising them, HIV leaves the host vulnerable to acquire several infections.

To replicate, the HIV virus uses the cell it infects. The process consists of several steps; these are visually depicted in Figure 7.1, and are now briefly described.

1. **Binding and Fusion.** The process begins when an HIV virion attaches or binds itself to the host cell using a gp120 protein found on the surface of the virion. Once attached, it fuses with the host cell and inserts its own genetic material into the cell.

2. **Reverse Transcription.** The HIV virus is a retrovirus, meaning it consists of RNA instead of DNA. In order for it to insert it's genome into the host cell's DNA, the RNA must first be transformed to DNA. This is done with the help of a viral enzyme called the *reverse transcriptase*.

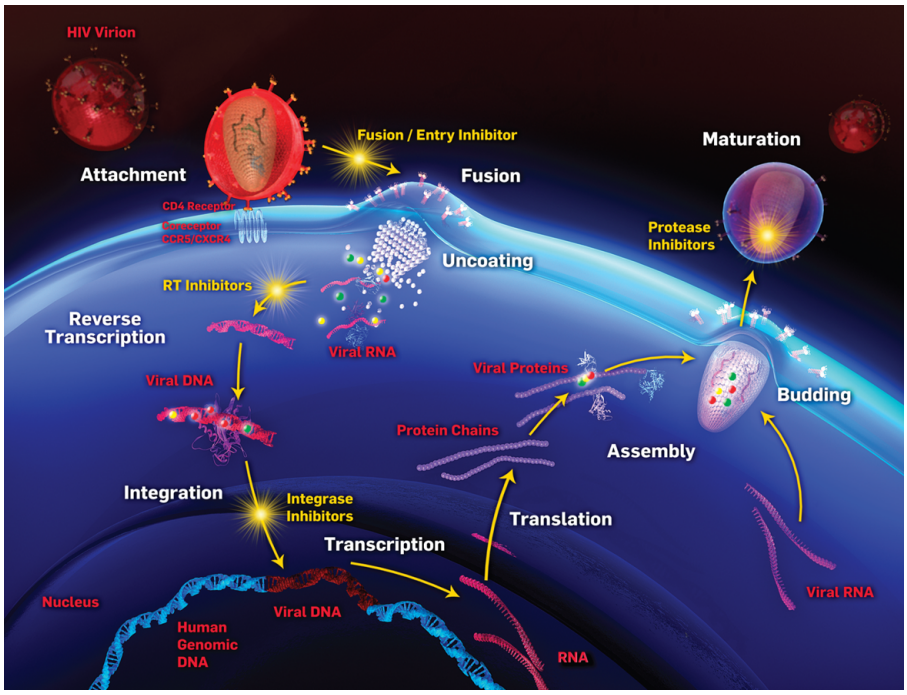


Figure 7.1: The life-cycle of the Human Immunodeficiency Virus. The image is reproduced from [LATK10]. The process begins when an HIV virion binds and then fuses itself to the host cell. To insert its genetic material into the host's DNA, reverse transcription takes place to transform the virion's RNA to DNA. In the integration phase, the newly transformed viral DNA is integrated with the host's DNA. Copies of the virus are assembled with the help of a viral enzyme called the *protease*. Finally, a new infectious virion buds from the cell.

3. **Integration.** Once the viral genome has been converted to DNA, it is transported to the nucleus of the cell where it is integrated with the hosts cell's DNA. The integration process is done with the help of a second viral enzyme called the *integrase*.

4. **Transcription.** When the host cell becomes activated, it will begin to generate copies of itself. However, because the virus has inserted it's genome into the cell's DNA, it will now generate copies of the virus in the form of long protein chains.

5. **Assembly.** The copies of the virus are assembled near the cell surface, where a third viral enzyme called the *protease* cuts the protein chain into individual proteins, rendering the newly formed virion infectious.

6. **Budding.** The new infectious virion pushes its way out of cell, looking for a new host cell to attach to.

7.1.2 HIV Treatment

The primary challenge in developing a vaccine for HIV lies in the evolving nature of the virus. Essentially, the virus avoids attack from the immune system or other external agents by rapidly changing its form. By mutating, the HIV virus is able to go undetected, infecting cells at a fast rate, and further compromising the host's immune system.

Currently, no vaccine for HIV exists. Instead, antiretroviral therapy (ART) is used to slow the replication rate of the virus. ART consists in taking a combination of HIV fighting medications. These medications prevent the HIV virus from multiplying and destroying healthy CD4 cells. Antiretroviral drugs act by blocking certain stages of the HIV life-cycle. They can be classified according to the stage they act on as: Entry Inhibitors, Reverse Transcriptase Inhibitors, Integrase Inhibitors and Protease Inhibitors.

While ART has been effective in slowing HIV replication, it is not a cure for HIV. Furthermore, the positive effects of a regimen are only temporary. HIV reacts to the selective pressure applied by the antiretroviral drugs by mutating to a drug resistant form. Consequently, in order to avoid further damage to the immune system, physicians must change patients' ART regimens once it ceases to have effect.

A lot of effort has been made to understand the relationships that exist between antiretrovirals and the drug resistant mutations [BSW+02, DP03, DCG+08]. However, this domain also presents a temporal aspect which is of great interest. Aside from understanding the dynamics between the antiretroviral drugs and the mutations, it would also be beneficial to know *when* the mutations occur. In [HLRFÁR+13] a TNBN was used to model the temporal relationships of a select group of protease inhibitors and mutations. By studying the temporal relationships that exist between antiretrovirals and the drug resistant mutations, physicians could design better treatment plans that foresee probable future mutations, and

thus, act preemptively against them.

7.1.3 Challenges in Predicting Mutations

In addition to ART regimens, there are other selective pressures that act upon the HIV virus causing resistance mutations. In fact, the human immune system plays a crucial role in how the HIV virus evolves. Specifically, Human Leukocyte Antigens (HLAs) molecules decide which agents are considered invaders and should be destroyed. To avoid attack from the immune system, the HIV virus must mutate to a form unrecognizable by the HLAs present in the host's body. HLA molecules can have many variations within a same population, and while this is advantageous for the prevalence of the human race, it has also led to wide diversity of resistance mutations in the HIV virus.

In addition, as ART programs have become more widespread, the virus has adapted to the antiretrovirals. As consequence, cases of transmitted resistance are occurring more frequently, meaning resistance to antiretrovirals will be carried from human to human.

In effect, there are many selective pressures that may influence which resistance mutations appear in a host. Since drug therapies for HIV tend to vary between countries depending on several factors (e.g., socioeconomic factors), and the genetic makeup of a population will vary by region, the dynamics between antiretroviral drugs and resistance mutations for one population may not be the same as for another. As a result, a computational model learned with data from one region of the world may not be suitable for all regions. Unfortunately, data is scarce for many geographical regions, reducing the possibility for learning a reliable model. In the experiments that follow, the algorithm discussed in this thesis is applied to learn the temporal relationships that exist between protease inhibitors and mutations in Europe by transferring information from a model built for the United States.

Protease Inhibitor	Mutations	Week of Appearance	Subtype	Country
IDV, LPV, NFV	L63P, L90M	24	Subtype B	United States

Table 7.1: An example of a record obtained from the Stanford HIV Drug Resistance Database. In this example a patient received the protease inhibitors IDV (Indinavir), LPV (Lopinavir) and NFV (Nefinavir), resulting in the resistance mutations L63P and L90M, 24 weeks later. The virus presented in the patient is subtype B, and the patient is originally from the United States.

7.2 Data Set

Data for the experiments was taken from the Stanford HIV Drug Resistance Database [RGK+03]. The experiments model the relationships between a select group of protease inhibitors and a group of mutations. Each record contains information on which protease inhibitors were applied, which mutations resulted and when they presented, the subtype of the virus and the country the host is from. An example of a record is displayed in Table 7.1.

7.3 Learning a Mutational Network for Europe

In this experiment, a mutational network for Europe is learned from a scarce group of records belonging to patients from European countries, and a larger group of records belonging to patients from the United States. The data from the Stanford HIV Drug Resistance Database was divided into two groups, patients belonging to a European country, and patients belonging to the United States. Subsequently, all patient records that presented a subtype different than subtype B were filtered out from both groups, resulting in a data set of 333 records for Europe, and 3381 records for the United States.

The intent of only using patient records belonging to one subtype is to increase the likelihood for the target and auxiliary domains to present underlying similarities. However, whether these two regions differ substantially or mildly was not previously known for this experiment.

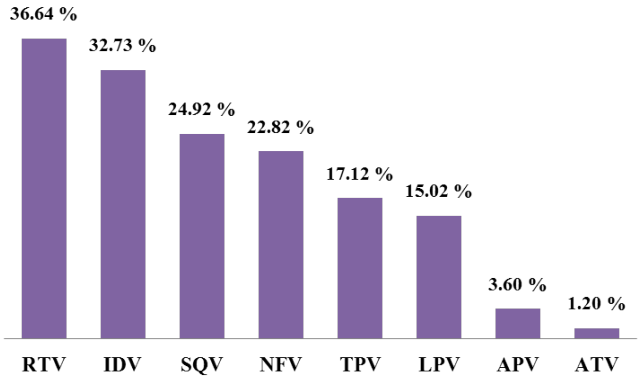


Figure 7.2: The histogram describes the frequency with which a group of protease inhibitors were applied to the patients in the data set for Europe consisting of 333 records.

In order to assess which antiretrovirals were applied more frequently, statistics were calculated from the 333 records belonging to the European countries. Figure 7.2 presents a histogram describing the frequency of application of the protease inhibitors. For the experiments, the five most frequent inhibitors were selected. Since the protease inhibitor RTV (Ritonavir) is only used as a booster in combination with other protease inhibitors, this inhibitor was omitted for this experiment, as the purpose is to model the relationships between protease inhibitors and mutations and not the relationships between protease inhibitors. The mutations modeled in this experiment were selected with the help of an expert in the field of HIV studies [1]. Table 7.2 presents a list of the protease inhibitors and the mutations used for this experiment.

In the following experiment the data set for Europe is partitioned into training and test sets. To evaluate the performance of the TNBN-TL algorithm, 4 different sizes of training sets were used to induce the mutational network for Europe (10, 33, 166 and 266 records). The experiments were repeated ten times to gain a better estimate of how the learned models behave, where for each repetition, a new training set was used for learning. Note that for training sets of sizes 166 and 266 the ten sets were not disjoint.

[1] Dr. Santiago Avila of the Research Center for Infectious Diseases in Mexico City provided his assistance for these experiments.

Protease Inhibitor	Mutation
Idinavir (IDV)	L90M
Saquinavir (SQV)	M46I
Nelfinavir (NFV)	V82A
Tipranavir (TPV)	A71V
Lopinavir (LPV)	L10I

Table 7.2: List of protease inhibitors and mutations used for learning a mutational network for Europe.

A partial order was provided where it was specified that mutations cannot be parent nodes to protease inhibitors, since it is quite obvious that a mutation is an effect produced by an antiretroviral and not the other way around. In addition, information was obtained from the field expert, where it was established that the protease inhibitors are never taken together. With this in mind, all protease inhibitors were made to be root nodes.

The results obtained by the TNBN-TL algorithm are compared to the results obtained by the TNBN-PC algorithm (no transfer) and with naive transfer. Specifically, two metrics are compared: the Relative Brier Score and the Time difference. For this experiment edit distances were not measured, as no gold standard exists to calculate this metric. Experimental results are compared using a T-test to determine statistical significance at the 95% confidence level. Table 7.3 presents the results of this experiment.

Results

From the results shown in Table 7.3 it is observed that the TNBN-TL algorithm obtains better results than the TNBN-PC algorithm for all metrics, in all cases. These results provide evidence that transfer learning is indeed improving the learning of a mutational network for Europe, as the TNBN-PC algorithm uses only the target data to induce the network. Even when 166 of the total target records are used for training (approximately 50%), the TNBN-

Data Set Size	Metric	TNBN-PC	Naive Transfer	TNBN-TL
10	RBS	53.19 (5.36)	**75.91 (3.56)** $*$	67.01 (12.12) †
	Time Difference	0.7 (0.11)	**0.33** (0.03)	0.41 (0.13) †
33	RBS	63.31 (5.68)	76.62 (0.81)	**79.84 (0.69)** †$*$
	Time Difference	0.48 (0.07)	0.32 (0.005)	**0.28 (0.005)** †$*$
166	RBS	69.68 (3.25)	75.36 (2.84)	**80.76 (0.84)** †$*$
	Time Difference	0.37 (0.04)	0.32 (0.01)	**0.28 (0.008)** †$*$
266	RBS	67.04 (4.79)	76.09 (2.73)	**81.24 (1.7)** †$*$
	Time Difference	0.41 (0.05)	0.31 (0.03)	**0.28 (0.01)** †$*$

Table 7.3: Comparison of the results of learning a mutational network for Europe with the TNBN-TL algorithm, the TNBN-PC algorithm and naive transfer. The standard deviations for the average of the evaluated metrics are displayed (σ). A '†' symbol expresses a significant difference between the TNBN-TL algorithm and the TNBN-PC algorithm, while an $*$ symbol expresses a significant difference between the results obtained by the TNBN-TL algorithm and naive transfer.

TL algorithm still obtains a RBS over 11% superior to the RBS obtained by the TNBN-PC algorithm. When using 266 target records for training (approximately 80%), the results reported for the TNBN-PC algorithm experience a drop; however, the difference with the results reported for 166 records is not statistically significant at the 95% confidence level, and this decrease is most likely due to a smaller testing set.

The majority of the results reported by the TNBN-TL algorithm also outperform those reported by naive transfer. However, the gap between the results is much smaller, and in fact, when using only 10 target records, naive transfer reports a superior RBS and measured time difference, though only the RBS is statistically significant. From the results obtained with naive transfer, it can be concluded that the domain for Europe (target domain) and the domain for the United States (auxiliary domain) do share strong similarities; however, while the two domains are similar, it cannot be stated that they are the same, since the results also show that the TNBN-TL algorithm outperforms naive transfer with data sets of 33, 166

and 266 records. This fact is an indication that the similarities that exist between the two domains are better leveraged by the TNBN-TL algorithm, and the negative impact of those aspects of the auxiliary domain that are more disparate is also reduced by the TNBN-TL algorithm. When only 10 target records are used for training, it is more difficult for the TNBN-TL algorithm to identify and leverage the existing similarities due to the extremely small data set.

Lastly, a TNBN model representing the mutational network for Europe was learned using transfer learning with the TNBN-TL algorithm using all 333 target records. Since no testing set existed, the model was validated with an expert in the field of HIV studies. The following is observed:

- The TNBN-TL algorithm successfully discovers the known relation between saquinavir (SQV) and the L90M mutation. This is the main resistance mutation associated with SQV.
- Strangely, idinavir (IDV) appears isolated, this could be do to a low amount of patient records using the inhibitor.
- Several mutations that tend to occur together are being identified by the model. For example, M46I and V82A confer resistance to Lopinavir (LPV) [MBN+02].
- When comparing the learned structure to the structure belonging to the model for the United States, an edit distance of 4 is reported. This value is relatively small in comparison to the number of arcs in the structures, indicating that both domains share many (in)dependencies.

Figure 7.3 shows the mutational network learned for Europe by transferring knowledge from data belonging to the United States. Variables representing protease inhibitors are modeled with blue nodes, while mutations are modeled with purple nodes. Relationships between mutations are represented with arrows using solid lines, and relationships between

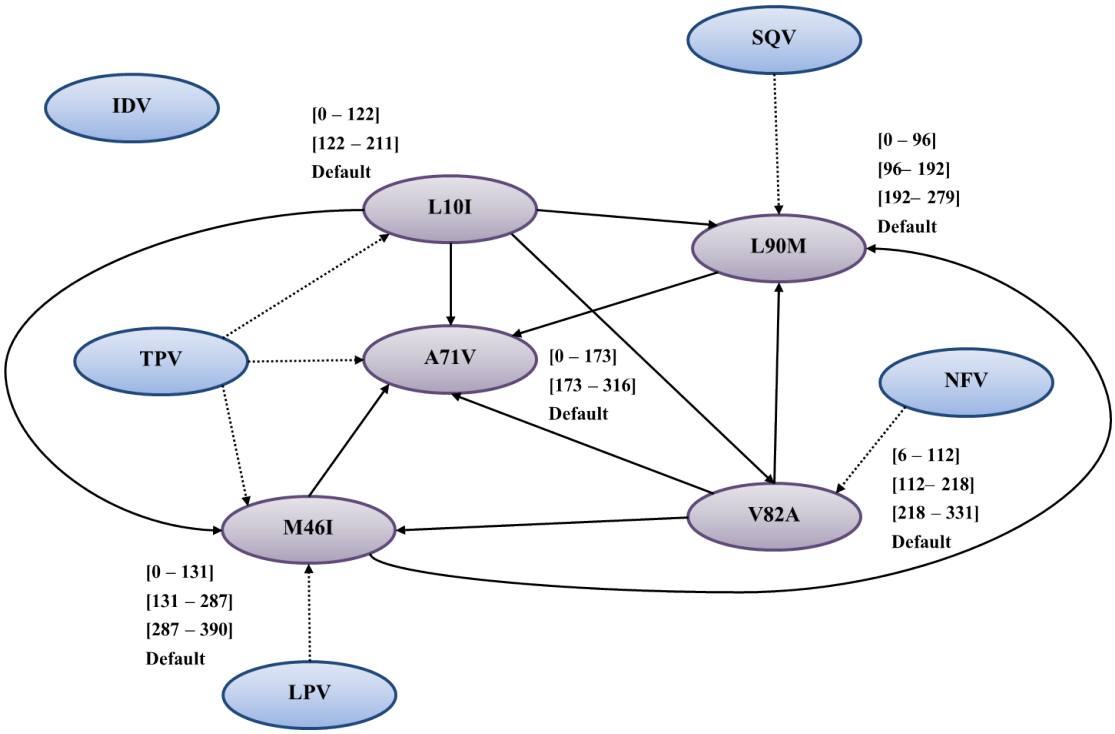


Figure 7.3: A mutational network for Europe learned from 333 target records, and by transferring from an auxiliary model for the United States. Protease inhibitors are modeled with blue nodes, while mutations are modeled with purple nodes. A solid arrow represents a temporal relationship between mutations, while a dotted arc represents a relationship between a protease inhibitor and a mutation. All protease inhibitors have boolean values. The temporal intervals indicating when the mutations appear are displayed next to their respective node.

protease inhibitors and mutations are represented with dotted line arrows. All protease inhibitors have boolean values. The temporal intervals for all mutations are displayed next to their respective node.

7.3.1 TNBN-TL Behavior Analysis

As with the synthetic collision domain, the predictive accuracy obtained by the TNBN-TL as a function of the number of available target data samples is analyzed. Again, the TNBN-TL algorithm showed a logarithmic deterioration rate for the measured RBS as less data samples

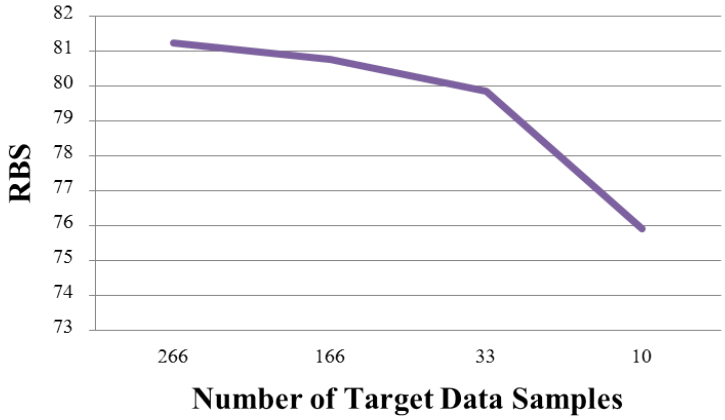


Figure 7.4: The behavior of the predictive accuracy obtained by the TNBN-TL algorithm as a function of the number of available target data samples for the medical domain of the Human Immunodeficiency Virus.

become available. Figure 7.4 shows this behavior.

7.4 Chapter Summary

In this chapter experiments done with real-world data for the medical domain of HIV treatment were presented. The experiments focused on learning a mutational network for patients from Europe presenting HIV subtype B. Scarce data was available for this region, and the TNBN-TL algorithm was applied to transfer knowledge from an auxiliary source domain belonging to the United States. For this experiment, models were evaluated experimentally, showing encouraging results for using transfer learning to compensate for the lack of training data for this real-world domain. The results obtained by the TNBN-TL algorithm in this domain give further evidence that this algorithm can be applied in other real-world domains where data for creating models is insufficient.

Chapter 8

Conclusions and Future Work

8.1 Summary

Temporal Nodes Bayesian Networks offer a representation for domains of dynamic nature while also keeping present the uncertainty that inevitably exists in their information. Aside from incorrect or incomplete data, insufficient data can also become an issue when learning models. In fact, scarce data sets make it particularly difficult to extract the meaningful patterns used for building reliable models, thus resulting in inaccurate predictions.

Transfer learning techniques offer a solution to the troubles brought on by scarce data sets by extracting useful knowledge from other different but closely related sources. In addition, transfer learning can also be applied to reduce the time it takes an algorithm to converge to a model. Transfer learning strategies for dynamic probabilistic graphical models like TNBNs have not been thoroughly explored. In this thesis, a transfer learning algorithm for inducing a Temporal Nodes Bayesian Network was proposed. To fully learn a TNBN, three components must be induced: the temporal intervals for the temporal nodes, the structure of the network and the parameters that quantify the model. For each component of the TNBN model, a learning strategy that employs knowledge transfer was defined:

- For learning the intervals a strategy that generates continuous Gaussian data from

the auxiliary domains was used. By generating continuous values, it is possible to compensate for the scarce temporal information available for the target domain.

- To learn the structure of the model, the PC-TL algorithm was adapted to use a node ordering. Since a temporal event requires a previous event to occur, the TNBN allows some restrictions on the ordering of the nodes to be made, and thus, the structural learning process can benefit from information on a partial order.

- Parameter learning was accomplished with two strategies. For instantaneous nodes the strategy proposed in [LSM10] for Bayesian networks was implemented, while for temporal nodes a new strategy that employs the Gaussian data generated for interval learning was developed.

Several experiments were performed to evaluate the proposed algorithm. Experiments were carried out both on synthetic and real-world data. In general, the following conclusions can be drawn:

1. When scarce amounts of training data exist, inducing an accurate structure can be difficult. By including a partial order for the PC-TL algorithm it is possible to reduce some of the negative effects of learning from scarce data sets, while also saving on computational resources.

2. Experiments show that incorporating information about the structure when learning the intervals results in a model that favors better predictive precision; however, the main benefit offered by interval refinement is the elimination of the need for explicitly stating the number of intervals for each temporal node. This information may not be easy to obtain, and an improper guess may lead to models that display poor results.

3. When auxiliary domains and the target domain are highly similar, the results obtained by naive transfer are comparable to the TNBN-TL algorithm which leverages the similarities. However, as more differences between domains exist, the TNBN-TL algorithm

obtains significant improvements over naive transfer, proving that it is identifying those elements which are most similar to the target domain, while reducing the impact of those that are more disparate.

4. Transfer learning significantly improves the models learned when scarce target data sets are available. When compared to two algorithms that do not apply transfer learning, the models obtained were significantly better when knowledge transfer was applied. These results give evidence that transfer learning can be employed to induce a TNBN in situations where scarce target data exists but similar auxiliary TNBNs are available.

5. The degree of similarity between an auxiliary domain and the target domain impacts how well the resulting models will perform. As the auxiliary domains become more closely related to the target domain there exist more similarities to leverage and less risk of negative transfer. However, since TNBNs consist of an additional component (the temporal component), to fully evaluate the strength of the relationship between two TNBNs, a similarity metric that considers the temporal aspects should be employed.

6. One limitation of the proposed algorithm is the requirement for the auxiliary domains to share the same variables as the target domain. This restriction limits the situations where the TNBN-TL algorithm can be applied, and the auxiliary sources it can transfer from. This problem can be resolved by mapping between domains with the help of a domain expert, or by defining an automatic mapping procedure.

8.2 Contributions

The main contribution of this thesis is an algorithm that transfers knowledge from auxiliary domains to induce a Temporal Nodes Bayesian Network, compensating for scarce training data. More specifically, the definition of this algorithm resulted in the following contributions:

- The definition of a procedure that employs transfer learning when inducing the temporal intervals associated to a temporal node.

- The adaptation of the PC-TL algorithm to use a partial order for structure learning. While experimental results reported better structure accuracy when using a node ordering, the main contribution of this adaptation lies in reducing the number of independence tests performed, thus saving computational resources.

- The definition of procedures that apply transfer learning to learn the parameters for the instantaneous nodes and the temporal nodes of a Temporal Nodes Bayesian Network.

- The application of transfer learning to a dynamic probabilistic graphical model, a field of study that has been underexplored.

It is also worth mentioning that the algorithm proposed in this thesis was applied to a real-world medical domain relating to the human immunodeficiency virus. Experimental results showed transfer learning to have a positive impact in the resulting models, providing evidence that this algorithm can be successfully applied in other real-world problems.

8.3 Future Work

From the observations made during the execution of this thesis, the following future work is suggested:

- The definition of a new similarity metric that considers the temporal aspects of the TNBN model. For this thesis the global similarity metric defined in [LSM10] was used to explicitly evaluate the strength of the relationships between the target and an auxiliary domain. By using this metric, an implicit assumption was made, that an auxiliary domain holding many common (in)dependencies with the target domain would also have very similar temporal ranges and intervals. This is not necessarily true, and the results reported in this thesis could improve by using a similarity metric better suited to the temporal aspects of the TNBN model. In addition, while the TNBN-TL algorithm is capable of retrieving a model even when the temporal ranges for target

and source domains are completely different, sufficiently substantial differences could lead to negative transfer thus resulting in a poor model.

- The adaptation of PC-TL to transfer from more than one auxiliary domain when inducing a structure. From the experiments performed with the synthetic collision domain (Subsection 6.3.3), it was observed that the structures obtained with naive transfer benefited from learning from multiple domains. It is possible, that by expanding the number of auxiliary domains that PC-TL transfers from, more accurate structures may be obtained.

- In addition, it would be interesting to apply transfer learning to a score and search structure learning method like the K2 algorithm. The K2 algorithm achieved better results with the synthetic data set as more records became available (Subsection 6.3.4). A comparison of the two types of structure learning algorithms could lead to interesting insights.

- Exploration of the possibility of using continuous distributions for temporal nodes instead of discrete temporal intervals. A continuous distribution would provide more precise estimations on the time point that the occurrence of an event takes place.

- Research in the area of domain mapping. In order to transfer, target and auxiliary domains must be expressed in the same manner. If the variables used to model the two systems are different, then knowledge transfer cannot occur unless a mapping between properties is achieved. The lack of a mapping procedure limits the situations where a transfer learning algorithm can be applied. For this thesis it was assumed that the auxiliary domains shared the same variables as the target domain.

- Lastly, it would be interesting to apply the TNBN-TL algorithm to more real-world domains so a further evaluation can take place. Aside from HIV, there are other medical domains where symptoms occur with a delay, and thus, these can be modeled with a TNBN.

References

[AFS99] Gustavo Arroyo-Figueroa and Luis Enrique Suear. A temporal bayesian network for diagnosis and prediction. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 13–20. Morgan Kaufmann Publishers Inc., 1999.

[AFSV98] Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Aljandro Villavicencio. Probabilistic temporal reasoning and its application to fossil power plant operation. *Expert Systems with Applications*, 15(3):317–324, 1998.

[Bri50] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[BSCR+83] Françoise Barré-Sinoussi, Jean-Claude Chermann, Fran Rey, Marie Therese Nugeyre, Sophie Chamaret, Jacqueline Gruest, Charles Dauguet, Charles Axler-Blin, Françoise Vézinet-Brun, Christine Rouzioux, et al. Isolation of a t-lymphotropic retrovirus from a patient at risk for acquired immune deficiency syndrome (aids). *Science*, 220(4599):868–871, 1983.

[BSW+02] Niko Beerenwinkel, Barbara Schmidt, Hauke Walter, Rolf Kaiser, Thomas Lengauer, Daniel Hoffmann, Klaus Korn, and Joachim Selbig. Diversity and complexity of hiv-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences*, 99(12):8271–8276, 2002.

[C+02] Elvira Consortium et al. Elvira: An environment for creating and using probabilistic graphical models. In *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 222–230, 2002.

[CH92] Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

[Chi96] David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.

[Cho02] Rizwan A Choudrey. *Variational methods for Bayesian independent component analysis*. PhD thesis, University of Oxford, 2002.

[CL68] C Chow and C Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[DCG+08] Koen Deforche, Ricardo J Camacho, Zehave Grossman, Marcelo A Soares, Kristel Van Laethem, David A Katzenstein, P Richard Harrigan, Rami Kantor, Robert Shafer, Anne-Mieke Vandamme, et al. Bayesian network analyses of resistance pathways against efavirenz and nevirapine. *Aids*, 22(16):2107–2115, 2008.

[DP03] Sorin Drăghici and R Brian Potter. Predicting hiv drug resistance with neural networks. *Bioinformatics*, 19(1):98–107, 2003.

[Dun73] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3(3):32–57, 1973.

[DXYY07a] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 210–219. ACM, 2007.

[DXYY07b] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 540. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[Fas10] Andrew Scott Fast. Learning the structure of bayesian networks with constraint satisfaction. *Open Access Dissertations*, page 182, 2010.

[fDCP81] Centers for Disease Control and Prevention. Pneumocystis pneumonia-los angeles. *Morbidity and Mortality Weekly Report*, 30:250–252, 1981.

[FG96] Nir Friedman and Moises Goldszmidt. Discretizing continuous attributes while learning bayesian networks. In *International Conference on Machine Learning*, pages 157–165, 1996.

[FY96] Nir Friedman and Zohar Yakhini. On the sample complexity of learning bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 274–282. Morgan Kaufmann Publishers Inc., 1996.

[HGC95] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[HLGMES13] Pablo Hernandez-Leal, Jesus A Gonzalez, Eduardo F Morales, and L Enrique Sucar. Learning temporal nodes bayesian networks. *International Journal of Approximate Reasoning*, 54(8):956–977, 2013.

[HLRFÁR+13] Pablo Hernandez-Leal, Alma Rios-Flores, Santiago Ávila-Rios, Gustavo Reyes-Terán, Jesus A Gonzalez, Lindsey Fiedler-Cameras, Felipe Orihuela-

Espina, Eduardo F Morales, and L Enrique Sucar. Discovering human immunodeficiency virus mutational pathways using temporal bayesian networks. *Artificial intelligence in medicine*, 57(3):185–195, 2013.

[HLSG+11] Pablo Hernandez-Leal, L Enrique Sucar, Jesus A Gonzalez, Eduardo F Morales, and Pablo H Ibarguengoytia. Learning temporal bayesian networks for power plant diagnosis. In *Modern Approaches in Applied Intelligence*, pages 39–48. Springer, 2011.

[HMG95] Steve Hanks, David Madigan, and Jonathan Gavrin. Probabilistic temporal reasoning with endogenous change. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 245–254. Morgan Kaufmann Publishers Inc., 1995.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[KF09] Daphne Kollar and Nir Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[KM08] Uffe B Kjrćulff and Anders L Madsen. *Bayesian networks and influence diagrams: a guide to construction and analysis*, volume 76. Springer, 2008.

[LATK10] Thomas Lengauer, André Altmann, Alexander Thielen, and Rolf Kaiser. Chasing the aids virus. *Communications of the ACM*, 53(3):66–74, 2010.

[LSM10] Roger Luis, L Enrique Sucar, and Eduardo F Morales. Inductive transfer for learning bayesian networks. *Machine learning*, 79(1-2):227–255, 2010.

[MBN+02] Bernard Masquelier, Dominique Breilh, Didier Neau, Sylvie Lawson-Ayayi, Valérie Lavignolle, Jean-Marie Ragnaud, Michel Dupon, Philippe Morlat,

F Dabis, H Fleury, et al. Human immunodeficiency virus type 1 genotypic and pharmacokinetic determinants of the virological response to lopinavir-ritonavir-containing therapy in protease inhibitor-experienced patients. *Antimicrobial agents and chemotherapy*, 46(9):2926–2932, 2002.

[Moo96] Todd K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[MSRS12] Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. Talmud: transfer learning for multiple domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 425–434. ACM, 2012.

[Mur02] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[NMC07] Alexandru Niculescu-Mizil and Rich Caruana. Inductive transfer for bayesian network structure learning. *Journal of Machine Learning Research-Proceedings Track*, 2:339–346, 2007.

[ODW01] Agnieszka Oniśko, Marek J Druzdzel, and Hanna Wasyluk. Learning bayesian network parameters from small data sets: Application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

[Org13] World Health Organization. *World health statistics 2013*. World Health Organization, 2013.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[PKY08] Sinno Jialin Pan, James T Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *Association for the Advancement of Artificial Intelligence*, volume 8, pages 677–682, 2008.

[Pre07] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[RGK+03] Soo-Yon Rhee, Matthew J Gonzales, Rami Kantor, Bradley J Betts, Jaideep Ravela, and Robert W Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic acids research*, 31(1):298–303, 2003.

[SGS00] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

[SH11] Paul M Sharp and Beatrice H Hahn. Origins of hiv and the aids pandemic. *Cold Spring Harbor perspectives in medicine*, 1(1), 2011.

[SS06] Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Association for the Advancement of Artificial Intelligence*, volume 6, pages 494–499, 2006.

[TK05] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI)*, pages 584–590, Edinburgh, Scottland, UK, July 2005.

[TKS08] Matthew E Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *Proceedings of the 7th international joint*

conference on Autonomous agents and multiagent systems-Volume 1, pages 283–290. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[TLR+12] Alberto Paolo Tonda, Evelyne Lutton, Romain Reuillon, Giovanni Squillero, and Pierre-Henri Wuillemin. Bayesian network structure learning from limited datasets through graph evolution. In *EuroGP*, pages 254–265. Springer Verlag, 2012.

[TSL05] Matthew E Taylor, Peter Stone, and Yaxin Liu. Value functions for rl-based behavior transfer: A comparative study. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 880. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[WHND11] Hua Wang, Heng Huang, Feiping Nie, and Chris Ding. Cross-language web page classification via dual knowledge transfer using nonnegative matrix tri-factorization. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 933–942. ACM, 2011.

[WLW12] Meng Wang, Wei Li, and Xiaogang Wang. Transferring a generic pedestrian detector towards specific scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, pages 3274–3281. IEEE, 2012.

[YJJ+10] Tianbao Yang, Rong Jin, Anil K Jain, Yang Zhou, and Wei Tong. Unsupervised transfer classification: application to text categorization. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1159–1168. ACM, 2010.

[ZCL+11] Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011.

Appendix A

Published Papers

The following publication resulted from this thesis:

- L. J. Fiedler Cameras, L. E. Sucar, E. F. Morales. A transfer learning approach for learning temporal nodes bayesian networks, in: The Twenty-Sixth International FLAIRS Conference, 2013.