UNED

Escuela Técnica Superior de Ingeniería Informática

Máster en Inteligencia Artificial Avanzada:
Fundamentos, Métodos y Aplicaciones

Probabilistic Graphical Models for the Tuning of Systems

Iñigo Bermejo

Advisor:
Prof. Francisco Javier Díez Vegas

Madrid - September 2012

Contents

List of Figures

Acknowledgements

I want to thank every person that has helped me during the making of this Master Thesis. Especially my advisor, Prof. Francisco Javier Díez, who has guided me throughout all the process, has helped me give shape to this Master Thesis and has been open for discussion over any subject concerning my research.

I would like to thank Prof. Paul Govaerts and the rest of members of the Opti-FOX project because a significant part of the work carried out in this Master Thesis has been motivated by the research problem they proposed us and by the many discussions we have had during the project. Our collaboration has been partially supported by a European Commission's 7th Framework Research for SME grant (7FP-SME 262266).

I would also like to thank the other members of the group I have been working with, Manolo, Manuel, Mar and Miguel for their help and patience.

I am grateful to Jesús Oliva, who let me use his Master Thesis as a model for mine and for his collaboration in the learning module of OpenMarkov.

I would like to thank Manfred Jäger for his guidance during my stay at Aalborg University and also Concha Bielza and Pedro Larrañaga for their useful comments on Interactive Learning.

In a more personal level, I would like to thank Clara as she is the one that has stood by me the longest, giving me support whenever I felt the need for it. Last but not least, I am very grateful to my parents and brother, whose constant support I feel even in the distance.

Abstract

Probabilistic Graphical Models (PGMs) have been widely praised for their declarative nature and their capability for complex reasoning with uncertainty, but when applied to real-world complex domains, the resulting model is usually large and highly inter-connected. This usually brings two main problems: first, the construction and maintenance of the model turns into a time-consuming, tedious and error-prone task. And second, the computational cost of inference soars with the number of links in the model. Therefore it seems necessary to come up with tools that will alleviate the issues that arise when dealing with large PGMs. In this Master Thesis we have proposed and implemented methods and techniques to help in the process of creation and maintenance of large PGMs.

Besides, we describe the process of modelling the problem of programming Cochlear Implants, *i.e.* adjusting parameters for their optimal performance with the use of PGMs. The new concepts and algorithms we have developed for this purpose are also presented in this Master Thesis. Even if inspired by the needs arisen throughout the development of this real-world application, they are valid for other domains, such as the tuning of systems with adjustable parameters.

Resumen

Los Modelos Gráficos Probabilistas (MGPs) han sido alabados por su naturaleza declarativa y su capacidad de razonamiento complejo con incertidumbre, pero cuando son aplicadas a problemas complejos de la vida real el modelo resultante es excesivamente grande y altamente interconectado. Esto suele acarrear dos grandes problemas: primero, la construcción y mantenimiento del modelo se convierte en una tarea costosa y tediosa. Y segunda, el coste computacional de la inferenci se dispara cuando el número de enlaces en el modelo crece. Por lo tanto es necesario crear herramientas que alivien los problemas que surgen cuando trabajamos con MGPs de tamaño considerable. En esta Tésis de Máster hemos proponemos e implementado métodos y técnicas que ayudarán a los ingenieros de conocimiento en el proceso de creación y mantenimiento de grandes MGPs.

Además, describimos la aplicación de los MGPs al problema del ajuste de parámetros en implantes cocleares. Los nuevos conceptos y algoritmos que hemos desarrollado, si bien han sido inspirados por la construcción del modelo mencionado, son aplicables en otros muchos casos, como por ejemplo la regulación de sistemas con parámetros ajustables.

Chapter 1

Introduction

1.1 Motivation

The Research Center on Intelligent Decision-Support Systems (CISIAD) of the UNED has specialized in the construction of Probabilistic Graphical Models (PGMs) and their application to different fields, mainly to medicine. Most of the contributions of this group have been motivated by real-world medical problems, and this Master Thesis is not an exception: the problem that has led our research is the programming of Cochlear Implants (CIs).

In Spain, around 2,500 babies are born annually with some degree of hearing loss and around 20% of them suffer from severe to profound deafness. Other people lose their hearing ability later as a consequence of genetic factors, infectious diseases (mainly meningitis), or other causes. A CI is an electronic device that allows people with severe and profound hearing losses recover a near normal hearing ability that permits them to integrate into society. Currently, there are over 200.000 CI users worldwide, with an annual increase of over 30.000. However, a CI has between 150 and 200 tunable parameters which makes its programming a difficult task. There are many CI users whose hearing performance is poor because of bad programming.

For this reason, several people have thought of the possibility of applying Artificial Intelligence (AI) to this problem. One of the main attempts is Opti-Fox, a project funded by a European Commission's 7th Framework Research for SME grant (7FP-SME 262266), which "aims at reducing the presence of experts when fitting cochlear implants to patients" [30]. Initially, this project intended to use data mining techniques to build a model for this purpose but the complexity of the problem, due to the huge number of variables involved, and the scarcity of data, due to a technical problem, made this approach unfeasible. Therefore, the leader of the project contacted the members of the CISIAD to study the possibility of building a PGM. The advantage of PGMs with respect to other AI techniques is that they permit to combine expert knowledge with data.

However, standard PGMs, such as Bayesian Networks (BNs) and Influence Diagrams (IDs) are not suitable for this task. This led us to develop a new type of PGM that we have called Tuning Networks. Because of the high number of variables in our model, with some repetitive structures, we have come to the conclusion that it would be a great advantage to apply an object-oriented approach. This way, the main contributions of this Master Thesis have been motivated by a real-world problem but are also applicable to other domains, not only in the field of medicine.

1.2 Objectives

The objetive of this Master Thesis is twofold.

On the one hand, the objective is to provide knowledge engineers with tools that will ease the task of creation and maintenance of large Probabilistic Graphical Models. For that purpose we present the design and implementation of an Interactive Learning module that will lead to the easier and faster creation of PGMs through structural learning with the help of the knowledge engineer or the expert. In addition, we present the implementation of a framework of Object Oriented Networks that will help knowledge engineers to model large PGMs by defining a hierarchy of objects and their relationships. Last but not least, we introduce the concept of Tuning Networks and describe the Tuning Model, that will help model the quite frequent domain of systems with tunable parameters.

On the other hand, the objective is to build a Probabilistic Graphical Model that will assist in the programming of Cochlear Implant devices, applying existing techniques as well as new ones presented in this Master Thesis. The author of this Master Thesis together with his advisor has taken part in Opti-Fox. The objective of our model will be to recommend the set of changes to the values of the parameters of the Cochlear Implant that will lead to the optimal performance of the device, which will be reflected upon the results of the tests that measure the patient's hearing ability.

1.3 Methodology

As mentioned above, the methodological contributions presented in this Master Thesis have been motivated by the need of new methods or adaptation of existing ones to solve problems that have arisen throughout the development of the CI fitting project. These are the steps followed in such cases:

- Review of the state of the art: Starting with a compilation of related literature usually guided by my advisor, involved a search for solutions to the problem faced or similar ones.
- Conceptual design: This phase consisted of developing new mathematical models and algorithms for the problems we wanted to solve.
- Implementation: It usually involved programming the new methods in OpenMarkov based on our earlier conceptual design, sometimes stepping back to the design or even to the analysis phase.

The methodology followed for the Implementation phase is the usual in following the standard phases of Software Engineering: specification, design, coding and testing. Also, OpenMarkov was developed with the Object Oriented Programming paradigm in mind. Therefore we have tried to design what we implement following its main principles, such as the SOLID principles.

All the new models and algorithms presented in this Master Thesis have been implemented in OpenMarkov. The author of this Master Thesis has been actively involved in the redesign and refactorization of existing functionality as well as the implementation of new features. Even if this additional work is not of direct use to the present Master Thesis, it will help others in the same way the existence of this open source tool has helped the author.

The collaboration with the company in charge of Opti-Fox has been based on regular meetings either in person (held in Antwerp, Belgium) or through Internet (with Skype). We have developed a knowledge engineer-expert relationship where the construction of the model has been a give-and-take between our knowledge of PGMs and their domain expertise.

1.4 Organization of the Master Thesis

The rest of the Master Thesis, is organized as follows. In the second chapter, State of the Art, a brief overview is done of the research areas the rest of the Master Thesis deals with, including a quick introduction to OpenMarkov, the software tool where the implementation work has been done.

The core of the Master Thesis lies in the third and fourth chapters, that contain the methodological contributions. The third chapter, Building Probabilistic Networks, contains the description of the tools and techniques proposed for the easier construction of PGMs and is itself divided in two blocks: Interactive Learning and Object Oriented Networks. The fourth chapter, Tuning Networks, describes a new type of PGM whose main feature is the extensive use of the tuning model, a new type of canonical model.

Chapter five describes the work done in the Opti-Fox project, namely the construction of the probabilistic model for the fitting of Cochlear Implants.

Finally, Chapter six, Conclusion, briefly reviews the main constributions and outlines the future work.

Chapter 2

State of the Art

2.1 Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) are a framework based on probability theory that models large problems involving many inter-related variables. The first and most widely spread type of PGMs are Bayesian Networks. Since Bayesian Networks were proposed by Judea Pearl in [25] PGMs have been widely praised as an effective framework for knowledge representation and reasoning under uncertainty.

A probabilistic graphical model consists of a joint probability distribution defined on a set of variables and a graph containing a node for each variable; the structure of the graph imposes some relations of conditional independence on the structure of the network, which depend mainly on the type of graph. Some types of PGMs are Bayesian networks (BNs), Markov networks, influence diagrams, hidden Markov models, factored MDPs and POMDPs, etc.

Notation

We will use capital letters to represent variables or nodes and lower case letters to represent the values of variables. For instance, v will represent a possible value of variable V. In the same way, $\mathbf{V}$ will denote a set of variables $\{V_1, \ldots, V_n\}$, and $\mathbf{v}$ a certain n-tuple $(v_1, \ldots, v_n)$, where v_i represents a value taken by variable V_i.

2.1.1 Basic concepts about graphs

A *graph* $G = (\mathbf{V}, \mathbf{E})$ consists of a finite set of nodes $\mathbf{V}$ and a finite set of edges $\mathbf{E}$. An edge is a pair of nodes (X, Y), where $X, Y \in \mathbf{V}$ and $X \neq Y$; if X and Y are ordered in the edge (X, Y) then it is said to be *directed*; otherwise it is *undirected.* A directed edge will be referred to as an *arc.* If every arc in $\mathbf{E}$ is directed then G is a *directed graph.* On the other hand, if every arc is undirected then G is said to be an *undirected graph.*

A *path* from a node X to a node Y in a graph $G = (\mathbf{V}, \mathbf{E})$ is a sequence $X = X_0, X_1, ..., X_n = Y$ of distinct nodes such that (X_{i-1}, X_i) is an edge in $\mathbf{E}$ for each i such that $1 \leq i \leq n$. The path is a *directed path* if (X_i, X_{i-1}) is a directed arc from X_{i-1} to X_i, for each i such that $1 \leq i \leq n$. A graph is said to be a *tree* if each pair of distinct nodes is connected by exactly one path.

A *cycle* is a path with the exception that $X_0 = X_n$, and a *directed cycle* is a directed path with $X_0 = X_n$. A directed graph with no directed cycles is said to be an *acyclic directed graph* (ADG).

Given an arc (X, Y) from X to Y, the node X is said to be a *parent* of Y and Y is a *child* of X. The set of parents for a node Y is denoted by $Pa(X)$, and the set of children for a node X is denoted by $Ch(X)$. The set of nodes from which there exists a directed path from X is named the *ancestors* of X (denoted by $an(X)$). Similarly, the set of nodes to which there exists a directed path from X is termed the *descendants* of X (denoted by $de(X)$).

2.1.2 Bayesian networks

A *Bayesian network* (BN) [25] $B = (G, P)$ consists of two elements: an ADG $G = (\mathbf{V}, \mathbf{E})$ in which each node $X \in \mathbf{V}$ (named *chance node*) is drawn as a circle and corresponds to a *chance variable* X; and a probability distribution over $\mathbf{V}$, $P(\mathbf{V})$, which can be factored as:

$$P(\mathbf{v}) = \prod_{X \in \mathbf{V}} P(x|pa(X)), \tag{2.1}$$

where $pa(X)$ denotes a configuration of the parents of X.

Since there is a bijection between a variable in a BN $B = (G, R)$ and a node in G, the terms node and variable will be used indifferently.

The quantitative information of a BN $B = (G, R)$ is given by assigning to each node $X \in \mathbf{V}$ a conditional probability distribution $P(X|Pa(X))$. Conditional probability distributions are also referred to as *potentials.* A *potential* is a real-valued function over a domain of finite variables. The *domain* $\phi = P(X|Pa(X))$ is $dom(\phi) = \{X\} \cup\ Pa(X) dom(\phi)$.

We will assume in the dissertation that BN $B = (G, P)$ is *discrete*, which means each chance node $X \in \mathbf{V}$ corresponds to a discrete *chance variable* X with a finite set of mutually exclusive and exhaustive states; the *domain* of a variable X is denoted by $dom(X) = (x_1, x_2, ..., x_l)$.

2.1.3 Influence diagrams

An *influence diagram* (ID) is basically a BN augmented with decision nodes and value nodes. Thus, an ID consists of an ADG $G = (\mathbf{V}, \mathbf{E})$, where the set $\mathbf{V}$ has three types of nodes: *chance nodes* $\mathbf{V}_C$, *decision nodes* $\mathbf{V}_D$ and *utility nodes* $\mathbf{V}_U$.

As in BNs, chance nodes (drawn as circles) represent chance variables, i.e., events which are not under the direct control of the decision maker. Decision nodes (drawn as rectangles) correspond to actions under the direct control of the decision maker. Utility nodes (drawn as diamonds) represent the expected benefit or loss, or more generally, the preferences of the decision maker. Utility nodes can not be parents of chance or decision nodes.

There are three types of arcs in an ID, corresponding to the type of nodes they link. Arcs into chance nodes represent probabilistic dependency. Arcs into decision nodes, named *informational arcs*, represent availability of information; i.e., if there is an arc from a node X to a decision node D then the state of X is known when decision D is made. Arcs into utility nodes represent functional dependency: arcs into ordinary utility nodes indicate the domain of the associated utility function.

We assume that there is a path in the ID that includes all the decision nodes, which induces a total order among the n decisions $\{D_1, \ldots, D_n\}$ and indicates the order in which the decisions are made. Such order originates a partitioning of $\mathbf{V}_C$ into a collection of disjoint subsets $\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_n$, where $\mathbf{C}_i$ contains every chance variable C such that there is an arc $C \to D_i$ but there is not an arc $C \to D_j$, $j < i$; i.e., $\mathbf{C}_i$ is the subset of chance variables known for D_i but unknown for any previous decision. This induces a *partial order* $\prec$ in $\mathbf{V}_C \cup \mathbf{V}_D$:

$$\mathbf{C}_0 \prec D_0 \prec \mathbf{C}_1 \prec ... \prec D_n \prec \mathbf{C}_n \ . \tag{2.2}$$

The set of variables known to the decision maker when deciding on D_j is termed the *informational predecessors* of D_j and is denoted $iPred(D_j)$function ψ_U of a utility node U can be expressed as a function of chance and decision nodes, termed the *functional predecessors*; the functional predecessors of an ordinary utility node being its parents.

The quantitative information that defines an ID is given by (1) assigning to each chance node C a conditional probability potential $p(C|pa(C))$ for each configuration of its parents, $pa(C)$[1], (2) assigning to each ordinary utility node U a potential $\psi_U(pa(U))$ that maps each configuration of its parents onto a real number.

A *stochastic policy* for a decision D is a probability distribution defined over D and conditioned on the set of its informational predecessors, $P_D(d|iPred(D))$. If P_D is degenerate (consisting of ones and zeros only) then we say that the policy is *deterministic*.

A *strategy* Δ for an ID is a set of policies, one for each decision, $\{P_D \mid D \in \mathbf{V}_D\}$. If every policy in the strategy Δ is deterministic, then Δ is said to be *deterministic*; otherwise Δ is *stochastic*. A strategy Δ *induces* a joint probability distribution over $\mathbf{V}_C \cup \mathbf{V}_D$ defined as follows:

$$P_\Delta(\mathbf{v}_C, \mathbf{v}_D) = P(\mathbf{v}_C : \mathbf{v}_D) \prod_{D \in \mathbf{V}_D} P_D(d|iPred(D)) = \prod_{C \in \mathbf{V}_C} P(c|pa(C)) \prod_{D \in \mathbf{V}_D} P_D(d|iPred(D)) \ . \tag{2.3}$$

We define the *expected utility of U under the strategy* Δ as $\mathrm{EU}_U(\Delta) = \mathrm{EU}_U(\Delta, \blacklozenge)$, where $\blacklozenge$ is the empty configuration. We have that

$$\mathrm{EU}_U(\Delta) = \sum_{\mathbf{v}_C} \sum_{\mathbf{v}_D} P(\mathbf{v}_C, \mathbf{v}_D)\, \psi_U(\mathbf{v}_C, \mathbf{v}_D). \tag{2.4}$$

We also define the *expected utility of the strategy* Δ as $\mathrm{EU}(\Delta) = \mathrm{EU}_{U_0}(\Delta)$. An *optimal strategy* is a strategy Δ_{opt} that maximizes the expected utility:

$$\Delta_{opt} = \arg\max_{\Delta \in \Delta^*} \mathrm{EU}(\Delta), \tag{2.5}$$

where Δ^* is the set of all the strategies for I. Each policy in an optimal strategy is said to be an *optimal policy*. The *maximum expected utility* (*MEU*) is

$$MEU = EU(\Delta_{opt}) = \max_{\Delta \in \Delta^*} EU(\Delta). \tag{2.6}$$

2.2 Building Probabilistic Networks

There are two main approaches for the construction of PGMs. The most common one is to build the model manually and it usually involves the collaboration of the knowledge engineer and the expert in the domain the PGM will represent. First, a graph is built based on the

[1]We denote with $Pa(X)$ the set of parents of X, and with $pa(X)$ a configuration of the parents of X.

expert's domain knowledge where causal relations are used to draw the arcs of the graph. Once the graph is built, the numerical parameters, that is, conditional probabilities are obtained from the literature (for example, medical journals), from databases, or from experts' estimations. Canonical models help alleviate the task of eliciting the numerical parameters as explained next.

The second method to build PGMs is to do it automatically based on a database, which is also called learning, covered in the next section.

2.2.1 Canonical models

The construction of directed graphical probabilistic models, such as Bayesian networks [25] and influence diagrams [17], requires specification of many conditional probability distributions of the form $P(y|\mathbf{x})$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is the set of parents of a node Y in the network—see Figure 2.1. The set $\{Y\} \cup \mathbf{X}$ is called a *family*. When all the variables in a family are discrete, $P(y|\mathbf{x})$ can be expressed in form of a conditional probability tables (CPT), whose size grows exponentially with the number of nodes. In general, the numerical parameters are obtained from databases or assessed by human experts and, for this reason, it is usually difficult to build a CPT for a family having more than three or four parents.

One way of reducing the complexity of elicitation of numerical probabilities is to rely on so-called *canonical models*, which allow for building probability distributions from a small number of parameters. The term "canonical" is used because such models are elementary units used in the construction of more complicated models [25]. Different canonical models may coexist in any probabilistic network. For instance, in causal Bayesian networks that model real-world domains, it is not uncommon that a significant number of families interact through OR/MAX-models, a few through AND-models and the rest of the families do not correspond to any canonical model, which implies that their CPT must be explicitly given.

Canonical models are useful not only because they simplify the construction of probabilistic models (knowledge engineering), but also because they save storage space and computation time [10] and because they correspond to causal patterns that can be exploited to generate user explanations [19].

ICI models

One kind of probabilistic models are those based on the assumption of *independence of causal influence* (ICI). Noisy ICI models can be defined by introducing n auxiliary variables $\{Z_1, \ldots, Z_n\}$, as shown in Figure 2.2, such that Y is a deterministic function of the Z_is, $y = f(\mathbf{z})$, and the value of each Z_i depends probabilistically on X_i, as captured by the CPT $P(z_i|x_i)$. In most ICI models, the Z_is have a causal interpretation. However, we can just see

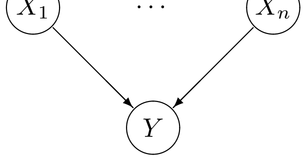


Figure 2.1: A family consisting of a child node Y and its parents, $\mathbf{X} = \{X_1, \ldots, X_n\}$.

them as auxiliary variables that are used for deriving the equations and are not part of the model. The conditional probability $P(y|\mathbf{x})$ is obtained by marginalizing out the Z_is:

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}} P(y|\mathbf{z}) \cdot P(\mathbf{z}|\mathbf{x}) \ , \tag{2.7}$$

where

$$P(y|\mathbf{z}) = \begin{cases} 1 & \text{if } y = f(\mathbf{z}) \\ 0 & \text{otherwise} \ . \end{cases} \tag{2.8}$$

Therefore,

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}|f(\mathbf{z})=y} P(\mathbf{z}|\mathbf{x}) \ . \tag{2.9}$$

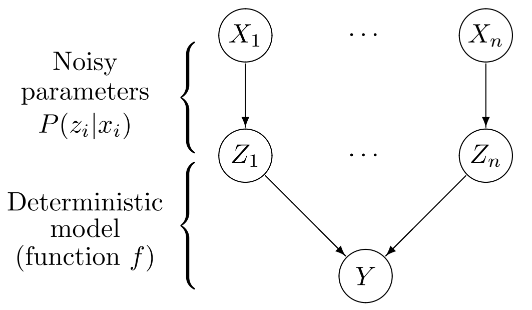

Noisy parameters $P(z_i|x_i)$
Deterministic model (function f)
X_1
X_n
Z_1
Z_n
Y

Figure 2.2: Auxiliary structure for the derivation of a noisy ICI model.

Independence of causal influence (ICI) means that there are no interactions among the causal mechanisms by which the X_is affect the value of Y. Given the graph in Figure 2.2, this property is equivalent to the absence of links $X_i \rightarrow Z_j$ and $Z_i \rightarrow Z_j$ for all $i \neq j$, which means that

$$P(\mathbf{z}|\mathbf{x}) = \prod_i P(z_i|x_i) \tag{2.10}$$

and, consequently,

$$P(y|\mathbf{x}) = \sum_{\mathbf{z}|f(\mathbf{z})=y} \prod_i P(z_i|x_i) \ . \tag{2.11}$$

Each parameter $P(z_i|x_i)$ of a canonical model is associated with a particular link $X_i \rightarrow Y$, while each parameter $P(y|\mathbf{x})$ in a CPT corresponds to a certain configuration $\mathbf{x}$ made up by all the parents of Y, and cannot be associated with any particular link. This property, stemming from the ICI assumption, entails two advantages from the point of view of knowledge engineering. The first is a significant reduction in the number of parameters required to specify a model, from $O(\exp(n))$ in a general model to $O(n)$ in a canonical model. This can amount to a substantial reduction of the elicitation effort. For example, a binary node with 10 binary parents will have a CPT consisting of $2^{11} = 2,048$ numerical parameters. Adding one more node doubles this number to $2^{12} = 4,096$ parameters. In contrast, a noisy OR model would require only 10 and 11 parameters, respectively. The second advantage is that the parameters in canonical models lend themselves to fairly intuitive interpretations, which facilitates the task of eliciting them from human experts. As mentioned above, canonical models not only require fewer parameters than ordinary CPTs, but also their parameters are more intuitive and easier to estimate.

It is possible to define leaky ICI models, which only differ from their noisy counterparts in the addition of another auxiliary variable, Z_L, which represents the effect of the variables not explicitly represented in the model [10].

2.2.2 Learning

Learning consists on automatically building the model that more truly represents the reality of a particular domain based on the information stored in a database. As in manual construction, learning can be divided in two phases: the structural learning and parametric learning. Parametric learning consists of computing the conditional probabilities given by the structure of the network using the observed frequencies on the database. Structural learning tries to find the graph that best represents the probability distribution based on the frequencies in the database. It is not uncommon though, to come across with situations where the graph or structure has been built with the help of an expert and it is just the probabilities that need to be learnt.

It is worth mentioning that when learning PGMs such as Bayesian Networks, two different graphs could in fact be probabilistically equivalent, that is, they can represent the same independence relationships and the same probability distributions over their variables.

There are two main methods for building the graph of a BN from a database. The first one consist of detecting the probabilistic conditional independencies present in the database. The most famous algorithm of this type is the *PC algorithm* [27, 28]. The second method, called *search and score*, consists of performing a heuristic search through the space of possible structures, using a metric that measures how well each structure can represent the probability distribution of the variables in the database. Several metrics have been proposed in the literature: Bayesian (which include K2 and BDe as particular cases), cross-entropy, AIC, and MDL—see [6] for references. K2, the first algorithm of this type, performed a search by departing from a network without links and adding at each step the link leading to the highest score, provided that the score was positive [7]. The method that proceeds by examining one operation at each step (adding, removing, or inverting a link) is called *hill climbing.*

Learning Bayesian networks is one of the most important research areas in the field of BNs. Every year, around one third of the total publications in that area are related to automatic learning.

Interactive Learning

Interactive structural learning was proposed by Sucar and Martínez-Arroyo [29] as a means for human-computer collaboration in the search for the optimal network structure. The system proposed by Sucar initially builds a tree automatically and then allows the user to modify it by adding, removing or inverting arcs. The strength of the correlation between the variables connected by a link is denoted graphically by the width of the link. It also shows a score representing the quality of the model, which is inversely related with the complexity of the network and the distance between the probability distributions specified by the network and the data.

Later de Campos [8] encoded expert knowledge in the form of a set of restrictions that the learning algorithm had to satisfy. They used three types of restrictions: presence of a link, absence of a link, and partial ordering of the variables. They proved that, for several data sets, this approach improved the quality of the networks learnt.

2.2.3 Object Oriented Networks

PGMs have been successfully applied to several domains of application, but when facing a complex domain, the construction and maintenance of the resulting large PGMs turns into a tedious, time-consuming and error-prone task. For example, the only mechanism for ”code reuse” is manually copying a network fragment and pasting it somewhere else. This duplication has the same drawbacks as the analogous process in a programming task. If the original network fragment is modified, the knowledge engineer must go back and manually change all of the models that used it. These difficulties have been encountered and solved in the context of programming languages, primarily via the introduction of complex data types. Koller and Pfeffer [18] originally proposed the Object Oriented Bayesian Networks (OOBN) as a solution to this problem. Applying the concepts of the Object Oriented Programming paradigm to PGMs, they described an OOBN language to model complex domains in terms of interrelated objects. Bayesian Network fragments are used to describe probabilistic relations between the attributes of an object where the attributes can be objects themselves, allowing the hierarchical definition of a model. As in Object Oriented Programming, the OOBN framework advocate for the creation of a hierarchy of reusable objects in order to reduce the complexity of the construction of the model.

Later Bangsø and Wuillemin further developed the idea of OOBNs in [4], giving a more detailed description of the framework, closer to implementation details than to theoretical definitions. They also introduced the idea of Top-down construction of Bayesian Networks, thanks to which it is possible to instantiate a class even if its specification is incomplete. This is probably the framework our approach owes the most, as it is on top of this framework that we have built ours, adding other functionalities.

Object Oriented Networks (OON) are meant to be a generalization of Object Oriented Bayesian Networks to PGMs other than Bayesian Networks. There are examples of Object Oriented networks with other PGMs such as in [3], where an Object Oriented Influence Diagram is proposed, but the authors use a very limited part of the funcitonality the framework offers.

2.3 OpenMarkov

This project started in 2002 at the Department of Artificial Intelligence of the Universidad Nacional de Educación a Distancia (UNED), in Madrid, Spain. Its original name was Carmen [1], but in 2010 it was renamed as OpenMarkov.[2] We departed from our experience in the construction of Elvira [12],[3] an open-source tool begun in 1997 as a joint project of several Spanish universities, but everything in the new program was redesigned and the code of OpenMarkov was built from scratch. The language chosen to develop OpenMarkov was Java, mainly to make it multi-platform.

OpenMarkov is able to represent several types of networks, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, and decision analysis networks (DANs), as

[2]`www.openmarkov.org`.
[3]`www.ia.uned.es/~elvira`.

well as several types of temporal models: dynamic Bayesian networks, Markov processes with atemporal decisions (MPADs), MDPs, POMDPs, Dec-POMDPs, and dynamic LIMIDs—see [2] for definitions and references. Currently it can only evaluate Bayesian networks, influence diagrams, and MPADs. Each network type is defined by a set of constraints [2, Appendix A], which leads to the possibility of defining new types of networks easily by combining the existing constraints and, if necessary, by adding new ones. Constraints play also an important role in the learning of BNs, as we will discuss below.

There are three types of variables in OpenMarkov: finite-states, numerical, and discretized. A discretized variable has a finite set of states, each one having an associated numeric interval.

The graphical user interface (GUI) is very similar to those of other software tools for PGMs, especially to that of Elvira. It has two working modes: edition and inference. It has been designed for internationalization; currently messages can be displayed in English and Spanish.

For a comparison of OpenMarkov with other open-source packages, check Table 2.1. For further details, see OpenMarkov's web pages and wiki.[4]

	Weka	JavaBayes	Elvira	BNT	Riso	UnBBayes	**OpenMarkov**	BayesLine	PNL	BNJ	OBP
Bayesian networks	yes	yes	yes	yes	yes	yes	**yes**	yes	yes	yes	yes
Influence diagrams	no	no	yes	yes	no	yes	**yes**	no	no	no	no
Dynamic models	no	no	no	yes	no	no	**yes**	no	no	no	no
Programming language	Java	Java	Java	Matlab	Java	Java	**Java**	Java	C++	Java	Python
License	GPL	GPL	?	GPL	GPL	GPL	**EUPL**	LGPL	IOSL	GPL	GPL
User manuals	yes	yes	yes	yes	yes	yes	**no***	no	no	yes	yes
Developer manuals	yes	no	no	no	no	yes	**no***	no	no	no	no
Users list/forum	yes	no	no	yes	yes	yes	**yes**	yes	no	yes	yes
Developers list/forum	yes	no	yes	yes	yes	yes	**yes**	yes	no	yes	yes
Source HTML docs	yes	yes	yes	no	yes	yes	**yes**	yes	no	no	no
Version control	yes	no	yes	no	yes	yes	**yes**	yes	no	yes	yes
Bug tracker	yes	yes	no	no	yes	yes	**yes**	yes	no	yes	yes
Start	1993	1996	1997	1999	2000	2000	**2002**	2003	2003	2004	2006
Stopped	—	2001	—	2007	2004	—	—	2003	2005	2004	2007

Table 2.1: Open-source packages for Bayesian networks. Some of those tools can also build and evaluate influence diagrams. The URLs for these packages can be found in K. Murphy's list, at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`, except for UnBBayes (`unbbayes.sourceforge.net`) and OpenMarkov (`www.openmarkov.org`). A star in a cell means that this feature will be available in the near future.

[4]`www.openmarkov.org`, `wiki.openmarkov.org`.

Chapter 3

Building Probabilistic Networks

We have proposed and implemented two techniques to help in the process of creation of PGMs. The first method, Interactive Learning, is a valuable resource to start building a model from an existing database with the guidance of an expert that has already proven to be very useful. The second is an evolution of existing frameworks that apply the philosophy of Object Oriented Programming to the realm of PGMs, that alleviates the task of the construction and maintenance of large PGMs.

3.1 Interactive Learning

3.1.1 Introduction

Algorithms for learning Bayesian networks (BNs) behave as a black box that takes a database as an input and returns a network as the output. In contrast, OpenMarkov, our tool for probabilistic graphical models, includes the option to run the algorithms in a step-by-step fashion, presenting a ranked list of operations (such as adding, removing, or inverting links) the user can select, while allowing live edition of the BN throughout the learning process. Database preprocessing options allow the user to select the variables to be used, indicate how to discretize numeric variables and impute missing data. It is also possible to use a model network to guide the learning process in different ways. This functionality in OpenMarkov can be employed to learn BNs with partial expert knowledge, to debug new algorithms, and as a pedagogical tool.

The difficulty and tediousness of manual construction of Bayesian networks has led to an increasing interest for learning methods that can generate PGMs from databases automatically. This is the preferred approach when there is a large database with few or none missing values, and no causal knowledge. However, in many cases the size of the database does not allow to learn a PGM that accurately represents the conditional independencies existing in the domain of application. Practitioners of these methods often encounter that the PGM obtained contains some links that the expert considers as obviously spurious, but given that in general the algorithms perform as black boxes, it is difficult to determine to what degree those links are really supported by the data.

Another problem is that most learning algorithms do not return causal models. In the last decade the number of studies aimed at obtaining causal models from databases has grown exponentially (the UAI Conference held in Barcelona in July 2011 was a clear illustration of this phenomenon). However in many cases the problem does not lie in the algorithm but on the lack of information in the database: the only conclusion that can be drawn reliably from a set of data—provided that it is big enough and not biased—is the set of correlations that exist in the real world. These correlations rule out some causal models, but the number of models compatible with the data is usually very large. Many of those models clash with common knowledge of the experts, but it is not easy to feed that knowledge into automatic causal learning algorithms. For this reason, it would be useful to have interactive learning algorithms that propose a list of changes to improve the accuracy of the network but allow an expert to select only those that do not contradict his/her knowledge.

Secondly, interactive learning could be also of great interest for researchers and developers of new algorithms. An interactive learning tool able to show on a graphical interface the different actions that the algorithm is considering at each step and the scores assigned to them may be very useful to debug the algorithm, for example, by observing how a shift in some of the parameters leads to a different selection of actions.

Thirdly, interactive learning programs may have a high pedagogical value by allowing the students to know the actions that the algorithm has evaluated at each step and why it has selected each action. Then it is possible to run a different algorithm, for example with a different search strategy or a different metric, and observe why it selects different actions at each step.

For these reasons we decided to implement an interactive learning module on OpenMarkov, an open-source software tool for editing and evaluating PGMs. The interactive learning module includes a user-friendly graphical interface that allows to overcome all the above-pointed problems, with the corresponding benefit for experts, researchers, developers and students.

3.1.2 Options for learning BNs in OpenMarkov

In this section we describe the main options that OpenMarkov offers for learning BNs interactively.

Using a model network

OpenMarkov gives the user the option to use an existing network as a model for the one that will be learned. There are four options. The first is to use the model only to determine the positions of the nodes. When we learn a network from a database we can place the nodes on the screen by dragging them with the mouse, trying to minimize the number of links crossing one another; but if we learn another network from the same database (for example, using different options for the algorithm), we should drag replace the nodes again. OpenMarkov facilitates this task by placing the nodes in the same positions as in a network built previously.

The other three options are whether the algorithm can add, remove, or invert the links present in the model network. They are useful, for example, when we wish that the algorithm preserves all the links in the model network. The first option (i.e., using the model only to place the nodes) is incompatible with the other three, which are compatible with one another.

There are other uses of the model network, that we describe below.

Data preprocessing

Unfortunately data in the databases is usually not suitable to be directly fed to the learning algorithm and has to be preprocessed. OpenMarkov offers the following options.

Selection of variables Usually raw databases contain information that is irrelevant for the model (e.g. the patient's name). OpenMarkov can learn a network that contains all the variables in the database, but it is also possible to tell it to use only those present in the model network. The third possibility is to select the variables one by one from a list.

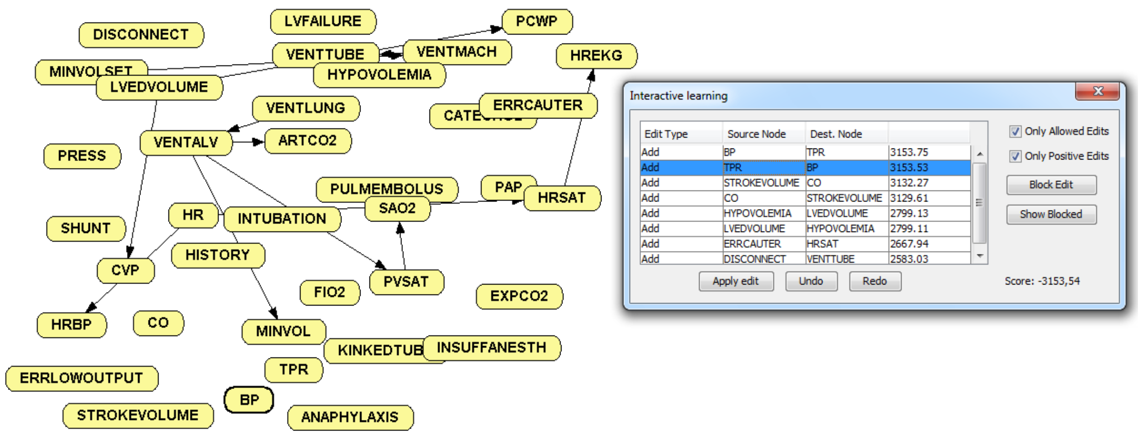
Discretization of numeric variables Currently OpenMarkov can only learn BNs with having finite states. Therefore, the numeric variables in the database must be discretized before the learning algorithm starts. OpenMarkov can discretize a variable in different ways. First, the user can indicate a number of intervals and then indicate whether the intervals must have the same width (considering the maximum and the minimum for that variable in the database) or the same frequency (i.e., the number of database registers for every interval will be the same). Second, if the variable is discretized in the model network, its intervals can be used to assign each number in the database to a state. For example, if a variable has three states, "negative", "null", and "positive", with three associated intervals, $(-\infty, 0)$, $[0, 0]$, and $(0, +\infty)$, respectively, these intervals can be used to discretize the values in the database. This way, creating a model network is a way of specifying how numeric variables should be discretized.

Imputation of missing values Currently OpenMarkov offers only two ways to fill in the gaps in the database: either to ignore every register that contains at least one missing value, or to write the value "missing" in every empty cell, which is then treated as if it were an ordinary value.

List of suggested edits

In OpenMarkov an *edit* is an atomic modification of a data structure. There are three edits that an interactive learning algorithm can propose: adding, removing, or inverting a link. The list is composed by sorting the edits according to their scores. The hill climbing algorithm computes the scores using the metric selected by the user. The PC algorithm performs many statistical test in which the null hypothesis is that two variables are not correlated given other variables. Roughly speaking, a high p resulting from the test suggests that two variables are conditionally independent, i.e., that a link can be removed. Therefore, the p value can be used as a score to rank the edits, each edit being the removal of a link.

Interactive learning is performed by having two windows: one showing the graph of the network and another one showing the proposed edits. The user can select any edit from the list, not necessarily the one having the highest score, and the change will be immediately displayed on the network window. Alternatively, the user can add or remove any link from the graph. In both cases, the scores will be recalculated and a new list will be proposed. Figure 3.1 shows the lists of suggested edits shown during the interactive learning process.



Edit Type	Source Node	Dest. Node	
Add	BP	TPR	3153.75
Add	TPR	BP	3153.53
Add	STROKEVOLUME	CO	3132.27
Add	CO	STROKEVOLUME	3129.61
Add	HYPOVOLEMIA	LVEDVOLUME	2799.13
Add	LVEDVOLUME	HYPOVOLEMIA	2799.11
Add	ERRCAUTER	HRSAT	2667.94
Add	DISCONNECT	VENTTUBE	2583.03

[✓] Only Allowed Edits
[✓] Only Positive Edits

Score: -3153,54

Figure 3.1: A moment of the interactive learning process: list of edits proposed by OpenMarkov and the network being learned.

Blocking edits

The user has the possibility of blocking an unwanted edit to prevent the system from offering it again and again. The list of blocked edits can be viewed and blocked edits can be later unblocked at any moment.

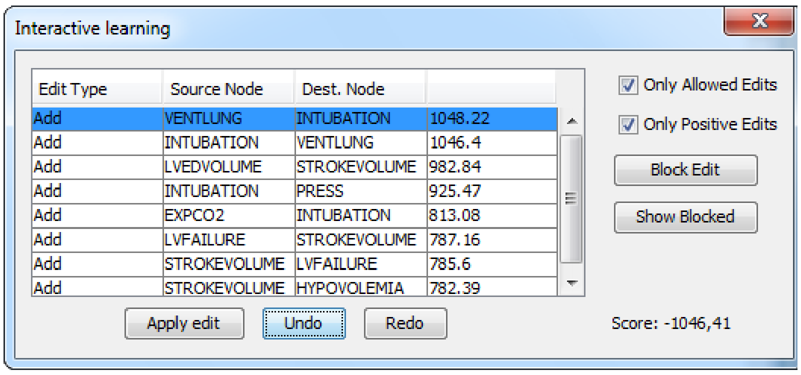
Additional options There are additional options to control the flow of the algorithm. One of them is to filter the edits proposed to those only those with a positive score. Another option is to show only the edits allowed by the constraints associated to the network. For example, a constraint stemming from the definition of the BN is that the graph cannot contain cycles. A constraint that the user can impose is that a node cannot have more than n parents. If the user selects the “Show only allowed edits” option, those incompatible with the constraints will not be shown in the list, even if they have a high score.

3.1.3 Case study

In order to explore the benefits of interactive learning, we study the case of learning the well-known ALARM network [5], which has 37 nodes and 46 links. The nodes are classified into three levels. The first level contains *diagnostic nodes*, which have no predecessors. The second level contains *intermediate variables*, representing pathophysiological anomalies that cannot be observed directly. The third contains *measurement nodes*, which represent clinical variables that can be observed or measured, and do not have children. The network contains no link from a lower level to an upper level.

Using this network we generated a database containing 10,000 samples and applied the hill-climbing algorithm with the K2 metric. We applied the learning algorithm automatically in OpenMarkov, resulting in a model with 50 links, 13 of which were not in the original network, even though 6 of them were inverted links of the original network. On the other hand, 9 of the original links were missing in the network learned.[1]

[1]The files used in this study are available at `www.openmarkov.org/learning` so that the results can be reproduced by other researchers.



Edit Type	Source Node	Dest. Node	
Add	VENTLUNG	INTUBATION	1048.22
Add	INTUBATION	VENTLUNG	1046.4
Add	LVEDVOLUME	STROKEVOLUME	982.84
Add	INTUBATION	PRESS	925.47
Add	EXPCO2	INTUBATION	813.08
Add	LVFAILURE	STROKEVOLUME	787.16
Add	STROKEVOLUME	LVFAILURE	785.6
Add	STROKEVOLUME	HYPOVOLEMIA	782.39

Score: -1046,41

Figure 3.2: The edit suggested at the top of the list contravenes our causal knowledge because VENTLUNG is an intermediate variable and INTUBATION is a diagnostic variable.

Then we learned the network interactively using elementary causal knowledge, according to which we did not accept the addition of any link from a measurement node to an intermediate or a diagnostic node, nor from an intermediate node to a diagnostic node. Figure 3.2 shows an example of a moment in the interactive learning process where the edit with the highest score cotravenes the causal knowledge. In this case, we chose to apply the second edit, which produces the same link but in the opposite direction.

The resulting net contained 47 links: only 2 of them were not in the original network and only one of the original links was missing. The two links “invented” by the learning algorithm were the last to be added and had such a low score that they might have been detected as spurious by the expert. We also observed that the missing link, from INSUFFANESTH to CATECHOL, has a very weak influence in the original network.

This experiment shows that even a portion of very rudimentary knowledge about the domain may lead to a significant improvement in the network built by our interactive learning algorithm.

3.1.4 Discussion

Advantages of our approach

As mentioned in the introduction, a problem of learning algorithms is that they often create spurious links due to small correlations existing in the database. Another problem is that in general the models obtained are not causal, not only because of the inherent limitation of most algorithms, but mainly because the information contained in the database does not permit to distinguish whether X is a direct cause of Y, or X is a cause of Y, or if there is a directed causal path between them involving other variables, or they have a common cause, or there is a selection bias in the database [15, 11].

OpenMarkov allows human users, who may be experts in their respective areas but novices in the field of probabilistic modelling, to supervise the execution of learning algorithms. The algorithm proposes some incremental modifications of the network, based on the information contained in the database, and the user has the opportunity to apply some of the changes proposed by the tool or impose others at any moment of the learning process, based on their expertise. Even if this might lead to a lower quality of the network according to the metric, the result might be better from the point of view of users’ acceptability, because human experts are reluctant to accept the advice of a machine if they cannot follow its reasoning [31].

An interactive learning tool might as well be useful for researchers that have developed a new algorithm and wish to trace its execution in order to debug or fine-tune it. This process can be done by inserting in the algorithm a few lines of code that print a trace on the standard output or in a file, but it is much nicer to observe graphically the operations performed by the algorithm, step by step, together with the qualitative information associated with the next modifications that the algorithm has evaluated. Obviously, this requires that the researchers implement the new functionality (such as a new metric, a new search technique, or a completely novel learning method). OpenMarkov's architecture has been carefully to permit these extensions: each new method can be implemented as a Maven subproject, that OpenMarkov will detect at run time as a plug-in.[2] This way, researchers can extend OpenMarkov dynamically without modifying the "official" source code.

Finally, an interactive learning program may be useful as a pedagogical tool to explain the performance of different algorithms: rather than observing the input and the output, students may follow the progression of the algorithm step by step, understanding why each change was selected, seeing the effects of taking different actions to those proposed by the system and comparing different algorithms.

Related work

Our work shares some aspects with the related work mentioned in the state of the art but presents important differences too. As in Sucar's paper [29], our proposal is also based on the idea of combining a learning algorithm and expert knowledge to build a model interactively. The main difference is that our system guides the expert through the creation process step-by-step instead of providing the final result of the learning algorithm and asking the expert to fine tune it. We think that our approach contributes to a better understanding of the behavior of the learning algorithm and therefore makes collaboration with the expert easier.

The idea of showing a score that represents the quality of the model is also present somehow in our approach, because the scores associated to each edit in the score and search algorithm represent the increment in the quality of the model, given by the complexity of the network and the distance between the probability distribution of the network and that of the data.

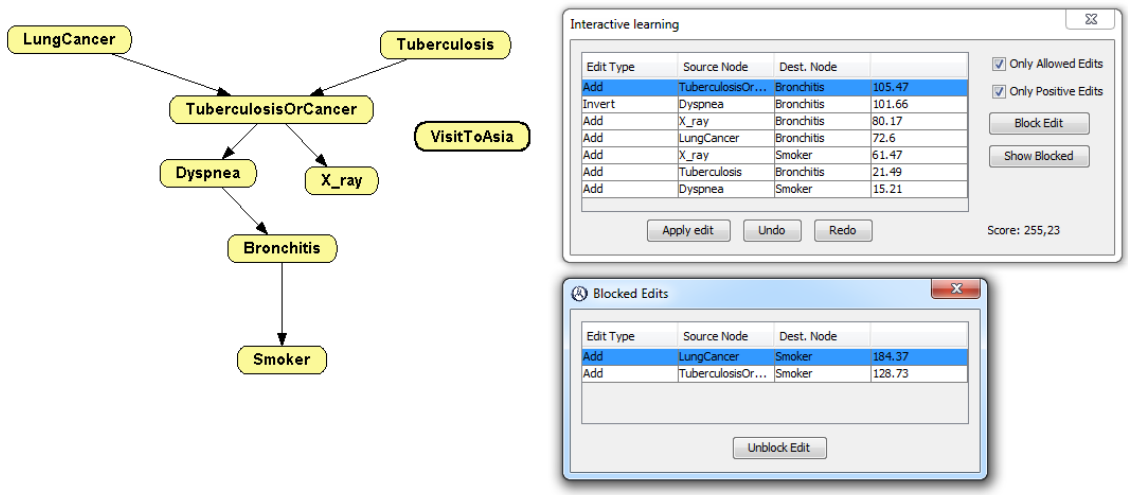
In the future, we will implement in OpenMarkov the possibility of representing the type of correlation (positive, negative, or null) by a color and the strength of the correlation by the thickness of the link, as we did in Elvira [21, 20], thus making our approach more similar to that of Sucar and Martínez-Arroyo.

Our work is also similar to de Campos' [29] proposal in that, by means of the model network, we can impose or prevent the presence of some links in the network learnt, but we do not have yet partial order restrictions. However, it would be easy to implement any restriction—not only those used by de Campos and Castellano—as an OpenMarkov constraint, as explained in Section 3.1.2. Another difference is that in OpenMarkov it is possible to specify whether the model network imposes the presence or absence of a link, or it only "suggests" them as a starting point, that can be overriden depending the scores computed by the algorithm.

[2]See `https://bitbucket.org/cisiad/org.openmarkov/wiki/OpenMarkov-organization.pdf`.

Another similarity between their work and ours is that both of them have been combined with score-and-search and independence-based algorithms. However, the main difference is that their approach is not interactive: their restrictions must be declared before the execution of the algorithm, which then runs automatically, while in OpenMarkov every action proposed by the algorithm can be accepted or rejected by the user, who can also impose any action at any moment of the process, thus giving full control to the user.

3.1.5 Design and Implementation



Edit Type	Source Node	Dest. Node	
Add	TuberculosisOr...	Bronchitis	105.47
Invert	Dyspnea	Bronchitis	101.66
Add	X_ray	Bronchitis	80.17
Add	LungCancer	Bronchitis	72.6
Add	X_ray	Smoker	61.47
Add	Tuberculosis	Bronchitis	21.49
Add	Dyspnea	Smoker	15.21

Score: 255,23

Edit Type	Source Node	Dest. Node	
Add	LungCancer	Smoker	184.37
Add	TuberculosisOr...	Smoker	128.73

Figure 3.3: A capture of the Interactive Learning module in OpenMarkov with the suggested edits and the Blocked Edits window

Requirements analysis

- User-friendliness or Usability: is the ease of use of the application. As any GUI based application, it need to be intuitive and efficient to use.

- Responsiveness: the ability of a system to complete assigned tasks within a given time. As it is going to be heavily interactive with a human, it needs to be fast.

Analysis model

The Interactive Learning module was designed to be a part of the Learning module in OpenMarkov. After refactoring the core of the learning module to offer the functionality needed by the interactive module (such as step-by-step execution), the implementation was included in the `org.openmarkov.learning.gui.interactive` package as the rest of the functionality was GUI related. The main class, *InteractiveLearningGUI* interacts with learning algorithms through the class *LearningManager* from the core of the learning module.

3.2 Object Oriented Networks

3.2.1 Introduction

PGMs have been successfully applied to medium-scale applications, but are inadequate as a general knowledge representation language for large and complex domains [24]. As Koller and Pfeffer put it in [18], "the construction of a network is a painstaking manual process, analogous to programming using logical circuits". Applying the paradigm of Object Oriented Programming to PGMs is an attempt to bring to the construction of PGMs the same advantages OOP offers over imperative programming: decoupling, abstraction, reusability, etc. Object Oriented Networks (OON) are meant to be a generalization of Object Oriented Bayesian Networks to PGMs other than Bayesian Networks.

3.2.2 Elements of OONs

The basic element of an Object Oriented Network is an object. An OON is defined by a set of classes, a set of instances of those classes, a set of nodes and a set of links. In the following we define each of these elements.

Classes

A class consists of a set of nodes and a set of links and it defines a probabilistic relation among its attributes. A class can also contain a number of instances(described later) of other classes. Any PGM can be considered a class that can be later instantiated. A class can have a set of input parameters: these parameters can be nodes or instances of other classes. When instancing these class, input parameters will need to be assigned values through reference links (explained later).

Instances

An instance is an instantiation of a class that has been previously defined. There can be several instances of the same class. Instances can be marked as input parameter; in this case, the instance is a placeholder to the real instance in the same way formal parameters are to actual parameters.

Nodes

Nodes can be chance, utility or decision nodes. As in the case of instances, they can be marked as input parameter in which case they act like placeholders. In [4] nodes marked as input parameters are called *reference nodes.*

Links

In addition to the common links used in PGMs, there is another type of links in OONs, referred to in [4] as *reference links.* These are directed links that match nodes or instances with input parameters of their same kind. When running inference, nodes and instances marked as input parameters will be replaced with those nodes and instances they are linked to through reference links.

3.2.3 Inference

The easiest way to run inference on a Object Oriented Network is to convert it first into a plain PGM, that is, replacing instances with the set of nodes and links they consist of and replacing the input parameters with the nodes and instances they are linked to through reference links. Once we have a regular PGM, inference is straightforward.

3.2.4 Parametric Learning

Another advantage of OONs is they are able to get better results from learning in object-oriented domains, as object-oriented learning takes into account the fact that all instances are identical as part of its learning bias. Langseth and Bangsø propose a learning method in [22] with and without missing data that learns in the class specification space instead of in each instance. That means every observation of a class instance will be considered as a case of the class. When the data is complete, Maximum Likelihood is used to learn from data and otherwise the Expectation Maximization (EM) algorithm is used. Therefore the only relevant difference with parametric learning in non object-oriented networks is that the updated potentials are those belonging to a class based on the records of the database referring to the instance of that particular class.

3.2.5 New elements in OONs

We have compiled a list of new elements we have added to the existing object-oriented frameworks:

Utility and Decision nodes

In Koller and Friedman's paper [18] as well as in Bangsø and Wuillemin [4], the only PGMs considered are Bayesian Networks. Our framework supports other important PGMs by supporting utility and decision nodes.

Multiple Arity parameters

In some situations, rather than having just one occurrence of an input parameter we have a variable number of them. For example, in a class representing a family, we could have a variable number of input parameters representing sons and daughters. This is what we have called multiple arity parameters, by which a variable number of reference links can point at a certain input parameter. This is restricted though to parameters whose links point at nodes with potentials supporting independence among parents, such as ICI potentials in the case of chance nodes or sum potentials in the case of utility nodes.

This is a similar idea to the one-to-many relationships used in Probabilistic Relational Models by Friedman et al. in [13], where they propose a framework for modelling PGMs based on the concept of relational databases.

Parameters in reference links

Reference links might be parameterized as to represent the strength of that particular link. For example, if we are trying to model the impact of several factors such as atmospheric pressure, humidity and temperature in weather, we could want to assign different strengths to different factors.

Other attributes in classes

There might be attributes in classes that are not represented by nodes but by a boolean or a double value. This attribute could be used as a parameter to render different values in the potentials of that class.

3.2.6 Design and Implementation

Requirements analysis

- User-friendliness or Usability: is the ease of use of the application. The task of the construction of Object Oriented Network is heavily based on the GUI and therefore it

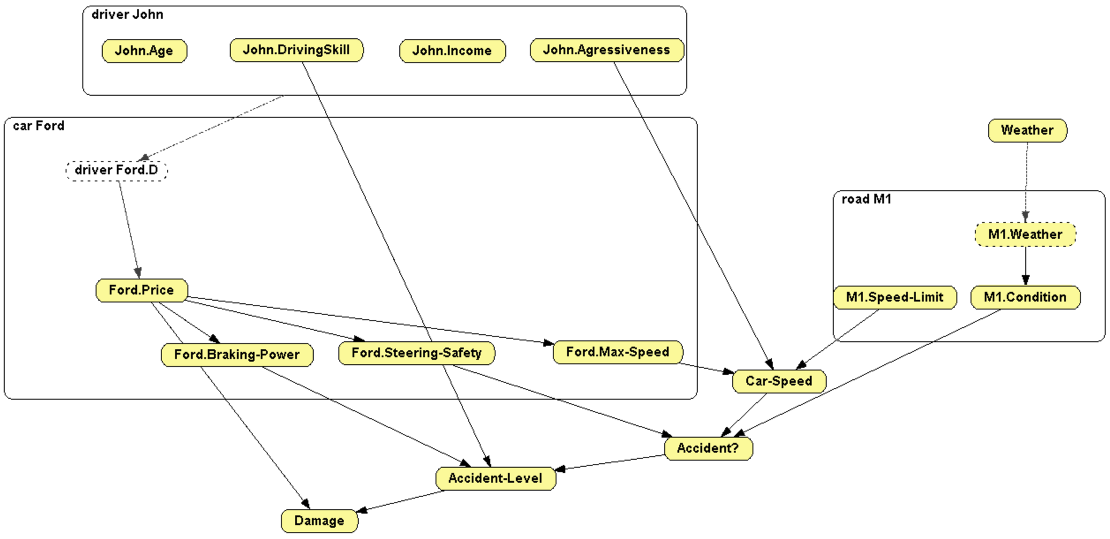


Figure 3.4: A capture of an Object Oriented Network model for a car accident, as seen in [18]

is important that it is intuitive and fast.

- Decoupling: it refers to the separation of software blocks that shouldn't depend on each other. In this case it meant that the implementation of OONs should be as decoupled as possible from the core and main GUI of OpenMarkov, since OONs are not a core functionality of OpenMarkov. It also meant users of OpenMarkov not interested in OONs should not be confused with OON related options in the main toolbar.

Analysis Model

Following, a description of the design of the elements implemented for Object Oriented Networks.

Instance Creation As part of the effort to decouple the implementation of OONs, the way in which "edition modes" in the GUI are handled was refactored. Edition modes were a static, hardcoded list consisting of: selection, (chance, utility and decision) node creation and link creation. After the refactorization, edition modes can be dynamically added implementing a subclass of `EditionMode`. Instance Creation is one of the new edition modes.

OON Toolbar In order to make instance creation easier, a toolbar was implemented which is only visible if activated in the *Toolbars* submenu under the *View* menu. The toolbar, as shown in Figure 3.5, contains two elements: a button to activate the Instance Creation mode and a combo box containing the list of classes to generate instances from. This list is fed by the list of current open networks in OpenMarkov. Therefore, in order to create an instance of a class, it first needs to be loaded. When Instance Creation is active, clicking on an empty spot of the network panel will generate an instance of the class currently selected in the class combobox.

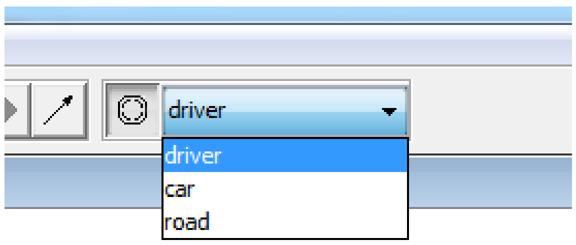


Figure 3.5: Object Oriented Network toolbar with a list of classes available to generate instances from.

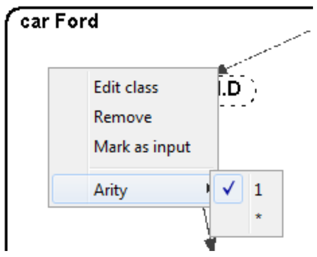


Figure 3.6: Contextual menu of an instance.

Instance Contextual Menu After an instance or more have been created, we might want to modify the class itself as part of the maintenance of the network. This can be easily done choosing the **Edit class** option in the contextual menu (shown in Figure 3.6) that is prompted when right-clicking on any instance of the class we want to edit. A network panel will be opened containing the class we want to modify. Any change done to the class network (addition or removal of nodes or links, changes to the potentials, etc.) will be automatically reflected on the existing instances.

The **Mark as input** option marks the selected instance as an input parameter. As described in Section 3.2, classes can have input parameters that can be either nodes or objects, that is, instances of other classes. Marking a node or instance as an input parameter tells the system that they expected a reference link pointing at them.

Finally, the **Arity** option lets the user set the arity of an input parameter, meaning it could hold one or many reference links. As explained in Section 3.2.5, in some situations there is a need for input parameters of an unknown arity, as when modelling a family, where the number of children is variable.

Chapter 4

Tuning Networks

We have defined a type of network that models the problem of finding the set of adjustments needed for the optimal performance of a system with tunable parameters.

4.1 Basic properties

4.1.1 Elements of the network

Tuning networks consist of an ADG and a probability distribution. Tuning networks can contain decision and utility nodes. Decision nodes represent tunable parameters in tuning networks. The values of these nodes can represent either setting the value of the parameter to a certain value in absolute terms or to do it in relative terms, that is, they represent the change in the value of the parameter (increase, decrease or no change). In both cases, we define the term of the *neutral state*, for relative variables as the "no change" value and for the absolute variables as the ideal value of the parameter.

Link Restrictions

A *restriction* associated to a link $X \to Y$, where X and Y are chance variables or decisions, is a pair (x, y), where x is a value of X and y is a value of Y. It means that variable Y cannot take the value y when X takes the value x. If X and Y are discrete, the restrictions associated with this link can be represented by a compatibility table with a column for each value of x and a row for each value of y; the cell (x, y) contains a 1 when x and y are compatible and 0 when there is a restriction. For example, Table 4.1 means that when the decision about blood test (BT) is not to perform it, the result (B) is neither positive nor negative.

BT		$+bt$	$\neg bt$
B	$+b$	1	0
	$\neg b$	1	0

Table 4.1: Compatibility table for the link $BT \to B$.

4.1.2 Interpretation of the network

A PGM representing the system described above has at least two posible interpretations:

The *diagnostic interpretation* assumes that the optimal tuning is unique, i.e., there is only one configuration of the parameters (the inputs) that makes the system perform optimally. The three values of each variable are interpreted as {*decreased*, *optimal*, *increased*}. The current output of the system is introduced as evidence into the Bayesian network and the goal of inference is to “diagnose” which parameters are not properly tuned. Therefore, inference proceeds down-up, i.e., from the observed outputs to the inputs. This method has two advantages: first, it does not require a global gain function, and second, inference is more efficient, because it evaluates the network only once, while the variational interpretation need to evaluate the network once for each change in one of the parameters.

The *variational interpretation* tries to *predict* the impact that a change in the value of the parents (the causes or the inputs) will have on the children (the effects or the outputs). The three values of each variable, $\{-, 0, +\}$ are interpreted as {*decrease*, *no-change*, *increase*}. Initially all the variables take on the value *no-change* by definition. The process of inference consists of computing the posterior probability of each output variable for each change that the user may impose on the input variables. Therefore, in this interpretation inference proceeds top-down, i.e., from (the possible adjustments of) the inputs to the outputs, using predictive reasoning. The changes that lead to an improved performance will be applied. When an improvement in some of the output variables comes together with a worsening in others, it is necessary to have a utility function that measure the global gain in performance.

4.2 The Tuning Model

The tuning model represents how a change in some variables (the parents) affects another variable (the child). The motivation for the proposal of this new model is that, when building a PGM for programming cochlear implants, we needed a model that could represent how a change in some of the parameters of the physical device affect other properties of the system, which in turn may affect the subject’s ability to detect and recognize the sound. Given that none of the existing models fitted our needs, we devised a new model based on the property of *independence of causal interaction* (ICI), which was discussed at length in Section 2.2.1.

4.2.1 Mathematical definition of the tuning model

As mentioned in Section 2.2.1, a noisy ICI model is defined by three elements: the domains of the variables, the function f, and some constraints on the values of $P(z_i|x_i)$.

In the tuning model, all the variables have the same **domain**, $\{-, 0, +\}$, where $-$ represents a decrease in the value of the variable, $+$ represents an increase, and 0 means “no change”. If the variable is denoted by V, we will sometimes write $v^-/v^0/v^+$ instead of $-/0/+$ to make it more clear what variable we are speaking of.

The **function** of the tuning model is defined as follows:

$$f_{\text{tuning}}(\mathbf{z}) = \begin{cases} y^+ & \text{if } n^+(\mathbf{z}) > 0 \\ y^0 & \text{if } n^+(\mathbf{z}) = 0 \\ y^- & \text{if } n^+(\mathbf{z}) < 0 \ , \end{cases} \tag{4.1}$$

where $n^+(\mathbf{z})$ is a function that returns the number of variables that take the value + in configuration $\mathbf{z}$ minus the number of those that take the value −. For example, $n^+(z_1^+, z_2^+, z_3^+) = 3$, $n^+(z_1^+, z_2^-, z_3^0) = 0$, and $n^+(z_1^+, z_2^-, z_3^-) = -1$. Therefore, $f_{\text{tuning}}(z_1^+, z_2^+, z_3^+) = y^+$, $f_{\text{tuning}}(z_1^+, z_2^-, z_3^0) = y^0$, and $f_{\text{tuning}}(z_1^+, z_2^-, z_3^-) = y^-$.

A **constraint** that we impose on $P(z_i|x_i)$ is that

$$P(z_i^0|x_i^0) = 1 \ , \tag{4.2}$$

which implies that $P(z_i^+|x_i^0) = P(z_i^-|x_i^0) = 0$. Therefore, if we introduce four parameters for each link, c_i^{++}, c_i^{+-}, c_i^{-+}, and c_i^{--}, the CPT for that link has the form shown in Table 4.2.

$P(z_i\|x_i)$	x_i^-	x_i^0	x_i^+
z_i^+	c_i^{-+}	0	c_i^{++}
z_i^0	$1 - c_i^{-+} - c_i^{--}$	1	$1 - c_i^{++} - c_i^{+-}$
z_i^-	c_i^{--}	0	c_i^{+-}

Table 4.2: Conditional probabilty table for link $X_i \rightarrow Y$ in the tuning model.

It is possible to prove from Equation 2.11 that when all the Xs take the value 0, then Y takes the value 0 with absolute certainty. When X_i takes the value + and the other Xs take the value 0, then Y takes the value + with probability c^{++} and the value − with probability c^{+-}. Similarly, when X_i takes the value − and the other Xs take the value 0, then Y takes the value + with probability c^{-+} and the value − with probability c^{--}. Therefore, the four c-parameters quantify the individual impact of X_i on Y.

4.2.2 Classes of interactions

We have seen that the general form of the conditional probability table associated with link $X_i \rightarrow Y$ is as shown in Table 4.2. However, it is possible to impose a second **constraint**: for each variable X_i and each value of this variable, $P(z_i^+|x_i) = 0$ or $P(z_i^-|x_i) = 0$; put another way:

$$(c_i^{-+}{=}0 \lor c_i^{--}{=}0) \land (c_i^{-+}{=}0 \lor c_i^{--}{=}0) \ . \tag{4.3}$$

Therefore, when this constraint holds for a link $X_i \rightarrow Y$, only two parameters are different from 0—in contrast with the general case, which requires four independent parameters—and that link must belong to one of four classes: direct, inverse, always increasing, and always decreasing.

The *direct* class is shown in Table 4.3. The values in the x_i^0 column are imposed by the first constraint of the tuning model (Eq. 4.2). The x_i^- column implies that a decrease in X_i causes a decrease in Y with a probability c_i^{--}, such that $c_i^{--} > 0$. It may occur, with probability $1 - c_i^{--}$, that a decrease in X_i fails to cause a change in Y, but that decrease can never cause an increase in Y. Similarly, the x_i^+ column means that an increase in X_i causes an increase in Y with a probability c_i^{++}. Therefore, this class represents a positive influence of X_i on Y [21] and a positive correlation between both variables.

Similarly, the *inverse* class is characterized by $c^{++} = c^{-+} = 0$, $c^{-+} > 0$, and $c^{+-} > 0$, which implies that a decrease in X_i causes an increase in Y, and vice versa, thus leading to a negative correlation between both variables.

$P(z_i \mid x_i)$	x_i^-	x_i^0	x_i^+
z_i^+	0	0	c_i^{++}
z_i^0	$1 - c_i^{--}$	1	$1 - c_i^{++}$
z_i^-	c_i^{--}	0	0

Table 4.3: Conditional probabilty table for a link $X_i \to Y$ of the *direct* class.

The relations that define the *always decreasing* class are: $c^{-+} = c^{++} = 0$, $c^{--} > 0$, and $c^{+-} > 0$. Therefore, any change in X_i will cause a decrease in Y. The properties of the *always increasing* class are analogous.

We may impose a third **constraint**: the *symmetry* of the influence. In the case of an *direct* interaction, it implies that $c_i^{++} = c_i^{--}$, i.e., the probability that an increase in X_i causes an increase in Y is the same as the probability that a decrease in X_i causes a decrease in Y. In the case of an *always decreasing* interaction, the probability of a decrease in Y is the same for an increase in X_i as for a decrease: $c_i^{--} = c_i^{++}$. A link satisfying the condition of symmetry requires only one parameter.

Several kinds of interaction may coexist within the same family.

Example 1. In a family with four parents, the link $X_1 \to Y$ might be general (i.e., free from the second and third constraints, as shown in Table 4.2), $X_2 \to Y$ might be a direct interaction, $X_3 \to Y$ might be direct and symmetric, and X_4 might be always decreasing. The total number of parameters for this model would be $4 + 2 + 1 + 2 = 9$.

If the interaction of this family did not use any canonical model, its conditional probability table would require $3^5 = 243$ parameters, but given that there are $3^4 = 81$ constraints among them, this family would require $243 - 81 = 162$ independent parameters. Obtaining those parameters from a database is unreliable, unless in the case of a huge database, because many of the configurations of the Xs will not be represented. Obtaining those parameters from an expert would be impossible in practice not only for the amount of time required, but mainly because estimating the probability of Y for each configuration of the Xs exceeds by far the cognitive capabilities of the human mind.

4.2.3 Causal interpretation of the tuning model

In the tuning model Y represents a parameter of a system whose value depends on the values taken on by other parameters, $\{X_1, \ldots, X_n\}$. Each auxiliary variable Z_i, associated with link $X_i \to Y$, as shown in Figure 2.2, indicates whether a change in X_i (from x_i^0 to x_i^+ or x_i^-) has induced a change in Y: z_i^+ indicates an increase (from y^0 to y^+) while z_i^- indicates a decrease (from y^0 to y^-).

The first constraint, given by Equation 4.2, means that when there is no change in X_i, then there is no change in Y.

The second constraint, given by Equation 4.3, means that a change in X_i may cause either an increase or a decrease in Y, but not both; this assumption seems reasonable for some domains, but there might be others in which an increase (or a decrease) in X_i sometimes produces an increase in Y and sometimes a decrease.

The function $f_{\text{tuning}}(\mathbf{z})$, given by Equation 4.1, means that when some of the X_is induce an increase in Y and others cause a decrease, the global effect depends on whether there are more increases than decreases, or vice versa, or there is a tie.

The tuning model is used to build probabilistic networks in which each variable represents a property of the system. In these networks, a node without parents represents a physical parameter that can be adjusted by the user, while a node Y with parents $\{X_1, \ldots, X_n\}$ represents a parameter or a property of the system whose value depends on other parameters (its parents).

4.2.4 Inference with the tuning model

Inference on the tuning model can be done either with exact or stochastic inference in quite different ways:

Exact Inference Using exact inference, conditional probabilities of the tuning model are obtained by marginalizing out the Z_is as in Equation 2.7. The computational cost of exact inference is $O(\exp(n))$ where n is the number of parents.

Stochastic inference In stochastic inference a sample is picked randomly for each variable. In order to pick a sample from the tuning model, one could first calculate the conditional probability as in exact inference and pick a sample from that probability distribution, but it is much more efficient to pick a sample for all the Z_is and then calculating $y = f(\mathbf{z})$, f function in Equation 4.1. The computational cost of stochastic inference is $O(n)$ where n is the number of samples.

4.3 Construction of a Tuning Network

The construction of a Tuning network for the tuning of a physical system begins by selecting the variables. Some of them will represent variations in the parameters of the system. The domain of each of these variables will be $\{-, 0, +\}$, which implies a discretization of a continuous variable. The value 0 might indicate that the value of the parameter has not changed at all, and $-/+$ might represent any increase/decrease in its value, no matter how small it might be. However, in practice it is better that 0 indicates "no significant change", $-$ represents a significant decrease and $+$ represents a significant increase. It is the knowledge engineer, in collaboration with human experts, who must determine what constitutes a significant change. For instance, the threshold might be $\pm 5\%$ of the absolute value of the parameter represented by the variable; for a different variable tuned with higher or lower precision the threshold might be $\pm 2\%$ or $\pm 10\%$, respectively This threshold might be different for each variable in the network, but it must be very clearly defined, because it will affect the elicitation of the conditional probabilities.

The second step in the construction of a Tuning network is to draw causal links between the variables, which is usually the easiest task in the construction of the network.

The third step is to analyze for each family in the Tuning network the possibility of applying a canonical model. The conditions for applying an OR, a MAX, a MIN, or an XOR model are discussed in [10], while the conditions for applying a tuning model have been described in the previous section.

The fourth step is to obtain the numerical parameters, i.e., the conditional probabilities for each family in the Bayesian network.

4.3.1 Elicitation of a tuning model

The first thing to do when considering to apply a tuning model to a relation in a family is to check whether the requirements are met. The first condition for applying a tuning model is that all the variables involved in the family must have the same domain: $\{-, 0, +\}$. The second condition is that the effects of the parents can be combined by applying the function f_{tuning} defined in Equation 4.1, which basically states that each change in one of the Xs may produce an increase or a decrease in Y, and the resulting value of Y depends on whether there are more increases than decreases, or vice versa, or there is a tie. The third condition is that the increase or decrease produced by each X_i only depends on the value taken by this variable, not on the values of the other Xs; this condition seems difficult to assess for a human expert because in fact the individual effects are combined by the function f_{tuning}, and consequently it is difficult to think of the individual effects "before" being combined. Therefore, it is reasonable to give the third condition for granted and assume that the tuning model can be applied whenever the first two conditions hold.

Once we know we can apply the tuning model, the next thing to do is to obtain the numerical parameters, i.e., the conditional probabilities for the family. In the tuning model, each link $X_i \to Y$ must be analyzed independently of the others. The first question is: "Does the second constraint, given by Equation 4.3, hold for this link, or is it possible that the same change in X_i sometimes causes an increase in Y and other times a decrease?" In the latter case it will be necessary to obtain four parameters: c^{++}, c^{+-}, c^{-+}, and c^{--}. However, some of the parameters might coincide; for example, using causal knowledge we might state that $c^{++} = c^{--}$ and $c^{-+} = c^{-+}$ (assumption of symmetry). This would reduce the number of independent parameters to be estimated.

On the contrary, if the second constraint holds, the next question to be asked is: "What class of interaction is this: direct, inverse, always increasing, or always decreasing?" (see Sec. 4.2.2). The last question is about symmetry. For example, in the case of a *direct* interaction, the question is: "Does a decrease in X_i cause a decrease in Y with the same probability that an increase in X_i causes an increase in Y?"If there is symmetry, we only need to elicit one parameter; otherwise, we need two.

Then, we have to estimate the numerical value(s) of the parameter(s) of each link. The question to be asked for each parameter can be derived from the comment in the last paragraph of Section 4.2.1. For example, the question for parameter c^{++} is: "What is the probability that an increase in X_i causes an increase in Y when there is no change in the other parents of Y?" The questions for the other parameters are analogous.

Finally, we must consider for that family whether a noisy tuning model suffices or it is necessary to apply a leaky tuning model.[1] The question is: "Is it possible that a change in some of the physical parameters not explicitly represented in the Bayesian network causes a change in Y?" If the answer is affirmative, the question: "What is the probability that they cause an increase in Y (when none of the explicit parents changes)?" will give us an estimate for the leak parameter c_L^+, will the question: "What is the probability that they cause a decrease in increase in Y?" will yield c_L^-, as

[1]In this paper we have not described explicitly the leaky tuning model due to space constraints, but it is analogous to the leaky models explained in [10].

4.3.2 Conditioned interactions

It may occur in practice that the effect of a certain variable—say X_1—on Y depends on the value of a third variable, C. For example, X_1 may represent a change in the value of a physical parameter (an increase or a decrease) while C represents the absolute value of that parameter. In this situation we can apply a modeling trick consisting of adding an auxiliary variable A_1, as shown in Figure 4.1. The interaction between A_1 and Y is given by the identity matrix: $P(a_1|y) = \delta_{a_1,y}$, where δ is Kronecker's delta function; put another way, the link $A_1 \rightarrow Y$ is a deterministic symmetric *direct* interaction (see Table 4.3) with $c_i^{--} = c_i^{++} = 1$.

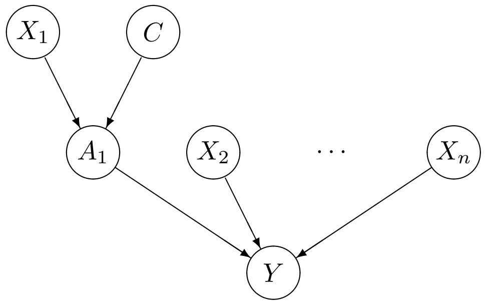


Figure 4.1: Conditioned interaction: the effect of X_1 on Y depends on the value of variable C. The combined effect of X_1 and C is modeled by the auxiliary variable A_1.

Table 4.4 shows with a hypothetical example how the effect of X_1 on Y may depend on a conditioning variable, C. When the value of C is *low*, the interaction is bottom-up and symmetric: in 90% of cases, a decrease in X_1 causes a decrease in Y, and vice versa. When C = *medium*, the interaction is also bottom-up and symmetric, but the effect is qualitatively smaller, i.e., it occurs in a lower proportion of cases. When C = *high*, the effect of a change in X_1 is unpredictable: it may cause an increase in Y but it may also cause a decrease, and the variability is asymmetric: it is higher for an increase in X_1 than for a decrease.

	C = *low*			C = *medium*			C = *high*		
$P(a_i\|x_i)$	x_i^-	x_i^0	x_i^+	x_i^-	x_i^0	x_i^+	x_i^-	x_i^0	x_i^+
a_i^+	0	0	0.9	0	0	0.4	0.04	0	0.26
a_i^0	0.1	1	0.1	0.4	1	0.6	0.68	1	0.67
a_i^-	0.9	0	0	0.6	0	0	0.28	0	0.07

Table 4.4: Conditional probability table for the auxiliary variable $A1$.

4.4 Inference: finding a near-optimal fitting

As in influence diagrams, inference consists on the search of an optimal strategy, in the case of tuning networks the set of adjustments or changes in the tunable parameter that will maximize the global expected utility of the model. As common inference algorithms for influence diagrams would be computationally unaffordable, we propose a greedy search algorithm that searches the space of possible strategies searching for the optimal strategy. The algorithm can be divided in two parts: score and search.

Search

The algorithm performs a greedy search over the space of possible strategies. The search space is limited by link restrictions, described in Section 4.1.1. The search is initialized setting all policies for all decision nodes to the *neutral state*. The greedy search then looks the single change in a decision node's policy (also referred to as an adjustment) that maximizes the utility. After changing the policy for that node, it repeats the search for the next change that will maximize the utility.

Score

The computation of the score is the computation of the global expected utility given a certain strategy. In this case, the computation of probabilities and utilities uses the same algorithms as influence diagrams.

Algoritmo 1: Search for near-optimal fitting

```
Data: Set of decision nodes, PGM
Result: Set of adjustments
set the imposed policies of all decision nodes to the neutral state;
find best adjustment;
while there is a better adjustment do
    add current best adjustment to result set;
    add current best adjustment to imposed policies;
    find best adjustment;
end
```

4.5 Implementation in OpenMarkov

4.5.1 Requirements Analysis

The requirements for the implementation of the Tuning Model were:

- Maintainability: is the ease with which a product can be maintained. Important factors that influence maintainability are keeping the code well documented and readable.
- Efficiency: optimization the speed and memory requirements of a computer program. Inference is a time-consuming process with tight time constraints. Memory efficiency is important too as memory needed for inference can increase exponentially on the number of variables.
- Robustness: is the ability of a computer system to cope with errors. In this case, with unexpected inputs.
- Reliability: the ability of a system or component to perform its required functions under stated conditions.

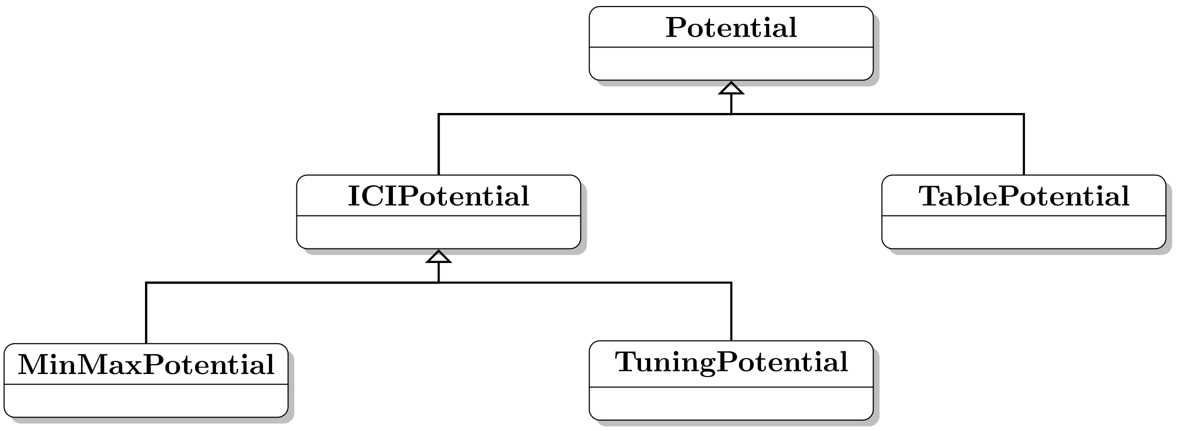
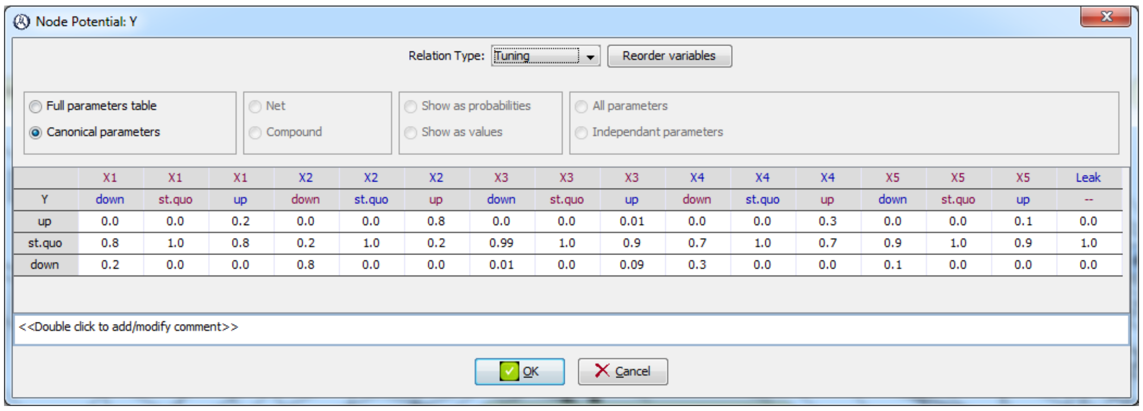


Figure 4.2: Part of the class hierarchy of Potential, where TuningPotenial appears as a subclass of ICIPotential

4.5.2 Analysis Model

The only class involved in the implementation of the tuning model is *TuningPotential*, which is a subclass of the abstract superclass ICIPotential. ICIPotential implements the generic behavior of ICI canonical models described in [10], and as earlier stated, the Tuning model is of the family of ICI canonical models. It is worth noting that the generic behavior of ICIPotential as well as its extensible interface were also designed and implemented by the author of this Master Thesis for the later implementation of the Tuning model. Thanks to this, the only thing to implement in TuningPotential was the functionality specific to the TuningModel, that is, to implement the function that retrieves the logic behind the f function in Equation 4.1. In Figure 4.2 the class diagram of the resulting class diagram can be seen.



Relation Type: Tuning

[✓ Canonical parameters] Full parameters table

Net / Compound

Show as probabilities / Show as values

All parameters / Independant parameters

	X1	X1	X1	X2	X2	X2	X3	X3	X3	X4	X4	X4	X5	X5	X5	Leak
Y	down	st.quo	up	down	st.quo	up	down	st.quo	up	down	st.quo	up	down	st.quo	up	--
up	0.0	0.0	0.2	0.0	0.0	0.8	0.0	0.0	0.01	0.0	0.0	0.3	0.0	0.0	0.1	0.0
st.quo	0.8	1.0	0.8	0.2	1.0	0.2	0.99	1.0	0.9	0.7	1.0	0.7	0.9	1.0	0.9	1.0
down	0.2	0.0	0.0	0.8	0.0	0.0	0.01	0.0	0.09	0.3	0.0	0.0	0.1	0.0	0.0	0.0

<<Double click to add/modify comment>>

Figure 4.3: A capture of the probability table of a tuning model with five parents in OpenMarkov

4.5.3 Design and implementation

Requirements Analysis

The requirements for the implementation of the Utility Calculator were:

- Decoupling: it refers to the separation of software blocks that shouldn't depend on each other. In this case it meant that the neither the core nor GUI of OpenMarkov should depend on the UtilityCalculator.
- Responsiveness: it refers to the specific ability of a system to complete assigned tasks within a given time. It also refers to the ability of using the system even if there is an on-going task.

The requirement of the decoupling was achieved thanks to the plug-in system implemented in OpenMarkov, thanks to which the UtilityCalculator works as a plug-in that is found and loaded in runtime. The UtilityCalculator is available under the Tools menu in OpenMarkov's main menu, but just when the jar produced is in OpenMarkov's classpath.

Responsiveness was mainly achieved by the use of approximate inference algorithms which take less time than exact inference algorithms. In order to improve responsiveness the Utility Calculator was implemented in such a way that the time-consuming task is run in the background so the GUI is not frozen and intermediate results are shown. For example, the best adjustment of each iteration is added to the table as soon as it is computed, while next iterations are being run in the background.

4.5.4 Utility Calculator: user manual

As previously explained when describing the Tuning networks, in order to find the optimal policies for the decision nodes, or to put it more clearly, in order to get the list of adjustments or changes in the parameters that lead to an optimal performance of the Cochlear Implant, a greedy search algorithm is run. A GUI was developed to give the user the possibility to run this algorithm and give some insight on its behavior.

The user introduces the available evidence using OpenMarkov's Network Panel and then launches the Utility Calculator. Next, the behavior of the main options of the GUI is described.

Utility Variable

The GUI lets the user choose the target utility node to be maximized among all the utility nodes in the Tuning Network.

Search type

The search algorithm can be run one iteration at a time (Step-by-step) or until the termination condition is met (Automatic), namely when no more tunings are found with a positive utility.

Tuning sequence

This table holds the ordered list of the best adjustments or tunings according to the algorithm and their relative utilities. The fields it contains are: Variable, referring to the tunable parameter that should be changed; State, defining the direction of that change; and Utility, holding the relative utility that will be summed to the total expected utility of the target utility variable if the change is applied. Once the search has started, this table is filled progressively as each iteration produces one best tuning. The total utility is also specified just below this table.

Next tuning

When running the algorithm in a step-by-step fashion, all the candidate tunings of the current iteration are shown in this table ordered by descending relative expected utility. While the algorithm adds the tuning with the highest relative utility to the list of the ”Tuning sequence”, the user can choose among these candidates by double-clicking on the corresponding row. This way the user not only has the chance to monitor the behavior of the algorithm but it can also change it. Once the user has chosen a tuning, it appears in the ”Tuning Sequence” table and the next iteration is executed, providing the ”Next Tuning” table with a new set of candidate tunings.

Run

This button launches the execution of the algorithm, be it on a step-by-step fashion or otherwise.

Apply Tuning Sequence

This button applies to the current network the Tuning Sequence that appears in the so called table. It is particularly useful when the list of recommended tunings is long.

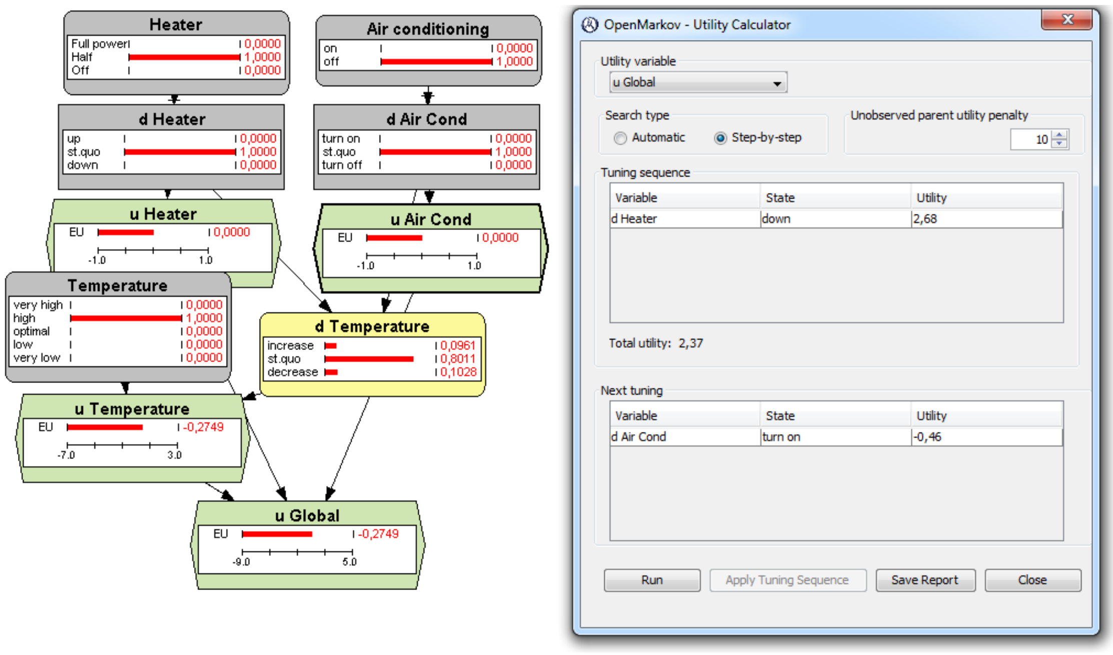
Save Report

This button gives the user the option to store the results of the Utility Calculator into an excel file.

4.6 Discussion

The tuning model arose from a need encountered when building a Bayesian network for real-world problem: the programming of cochlear implants. In this domain of application, it soon became clear that the diagnostic approach was inappropriate, hence we decided to use the predictive approach positive results.

The fact that the tuning model arose from a real-problem is a difference with some of the canonical problems proposed in the literature, which came out from mathematical speculation and have never been implemented on a software tool nor used in practice.



Utility variable: u Global

Search type: Automatic [✓ Step-by-step]

Unobserved parent utility penalty: 10

Tuning sequence

Variable	State	Utility
d Heater	down	2,68

Total utility: 2,37

Next tuning

Variable	State	Utility
d Air Cond	turn on	-0,46

Figure 4.4: A capture of a net modelling an air conditioning and heater system and the Utility Calculator's output. In this case, the Heater is switched on at half power,the Air conditioning is switched off and the Temperature is "high". The Utility Calculator recommends to switch off the Heater but does not recommend switching the Air conditioning on as this would have a cost that is bigger than the benefit expected from doing so.

Chapter 5

Application: Fitting of Cochlear Implants

5.1 Introduction

5.1.1 Cochlear Implants

Hearing is a complex function that converts sound (mechanical waves) into electrical patterns in the auditory nerve. The sound receptor is the inner ear, also known as the cochlea. Severe and profound hearing losses can be treated with cochlear implants (CI). A Cochlear Implant consists of a "speech processor" that analyzes the sound and an array of implanted electrodes that are stimulated to generate an electrical field that passes the information directly to the auditory nerve.

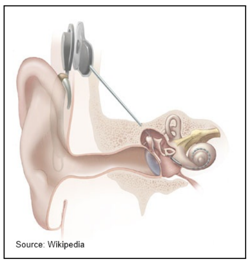


Figure 5.1: Cochlear Implant

A Cochlear Implant is controlled by about 200 tunable parameters that determine its behavior: sensitivity levels at different frequencies, electrical dynamic ranges for each electrode, characteristics of amplifiers and strategies of stimulating electrodes.

5.1.2 The difficulty of programming CIs

After implantation, CIs need to be programmed or "fitted" to optimize the hearing capabilities of each patient. This is usually a challenging and time-consuming task that is typically performed by highly trained audiologists or medical doctors. Recipients using inappropriate Cochlear Implant programming experience poor performance and outcomes.

CI centres and manufacturers have developed their own heuristics, usually in the for of simple "if-then" rules applied in a very flexible but individual and uncontrollable way.

5.1.3 The FOX and Opti-FOX projects

Otoconsult, in charge of the Opti-FOX project, had previously developed a similar application called FOX ([16], [32]), which is already in use in several centers across Europe, achieving decent results. FOX is a system based on parameterized deterministic rules and as such has its own limitations. Even if rule based systems are easy to implement from expert knowledge, as the rule database grows, so do conflicts between rules and the system gets harder to understand and debug. Besides, rule based systems are unable to learn from data.

Opti-FOX was conceived as an attempt to overcome FOX. In the beginning an approach with supervised classification algorithms such as the classifier k-NN, but failed to progress due to two main factors: the complexity of the problem (the number of parameters to be tuned is high) and the small number of records available.

Therefore, in order to improve the results of a system based on expert knowledge but limited reasoning power and a system that tried to reason over data mostly ignoring expert knowledge, the choice of PGMs seemed logic, as they combine the declarative modeling of expert knowledge and good reasoning abilities, including the ability to learn from data.

The goal is to build a Decision Support System (DSS) that will recommend a list of interventions or tunings to the parameters of the cochlear implant that will lead to the optimal performance of the implant.

5.2 Construction of the Model

Important notice: for the sake of confidentiality, some elements of the model have been highly simplified.

5.2.1 Variables in the model

The variables in the proposed model represent either tunable parameters of the CI device, the amount of energy in the audiory nerve and or the results of tests. Each electrode has a set of tunable parameters, some of which are:

- Lower threshold (T): Softest electrical input level detectable by user.
- Level of Maximum Confort (M): Electrical input level that is perceived as loud but comfortable to the user
- Pulse Width: Duration or length of one single phase

Besides, the Cochlear Implant (CI) has a set of tunable parameters that are electrode-independent, i.e. global to the implant. For example:

- Input Dynamic Range (IDR): Range of acoustic inputs
- Sensitivity: Determines gain of processor microphone, which subsequently determines the quietest sound picked up by the microphone

Each of these parameters is represented in the model by a chance variable representing the current value of the parameter and a decision node representing the relative change proposed.

The model also represents the result of a battery of different tests such as:

- Audiometry: identifies hearing threshold levels of an individual
- Phoneme Discrimination: measures the the ability to discriminate between pairs of phonemes
- Loudness Scaling: measures the ability to identify the loudness of sounds
- Speech Recognition: measures the ability to properly understand speech in different ambients.
- etc.

Each test is modeled with three different nodes: a chance node representing the current value of the test, a chance node representing the expected change in the result of the tests given a set of adjustments to the parameter and a utility variable modelling the expected utility based on the other two.

In order to have a global measure we can maximize, it is necessary to introduce some kind of weighted sum of the result of all tests. Utility nodes as introduced in influence diagrams are the perfect candidates for this, as they offer several models to combine the results of different outcomes.

5.2.2 Structure of the Model

There are two ways to describe the structure of the model. According to the first, the net consists of three horizontal layers where links go from one layer to the ones below it as shown in Figure 5.3. The top layer is the one formed by the tunable parameters of the Cochlear Implant device. The middle layer consists mainly of unobserved nodes and it reduces the high connectivity we would otherwise need between parameter and test nodes. Finally, the bottom layer contains the nodes representing the result of tests.

The other way of describing the structure is as the interaction between the set of electrodes and the set of frequency bands they are mapped to. In Figure 5.2 below we have chosen not to show the electrode-indenpendent parameters for the sake of clarity but it is assumed they influence all frequency bands. Electrodes affect the frequency band they are mapped to, as well as the neighboring frequency bands.

5.2.3 Elicitation of numerical parameters

The probabilities, utilities and link restrictions have been elicited by the expert. The probabilities are subjective estimates based on the expertise of the medical doctor leading this project. Utilities have been estimated by roughly assigning monetary value to each increase or decrease in the value of the ourcome variables. Finally link restrictions are based on common sense and physical limitations of CI devices.

5.2.4 Running Inference

Given the high number of variables in the network and the high number of parents some nodes have (there are a few with 9 parents), the computational as well as memory cost of running exact inference algorithms is unaffordable. For that reason, we had to implement an Approximate Inference algorithm to run inference on the network. After considering different

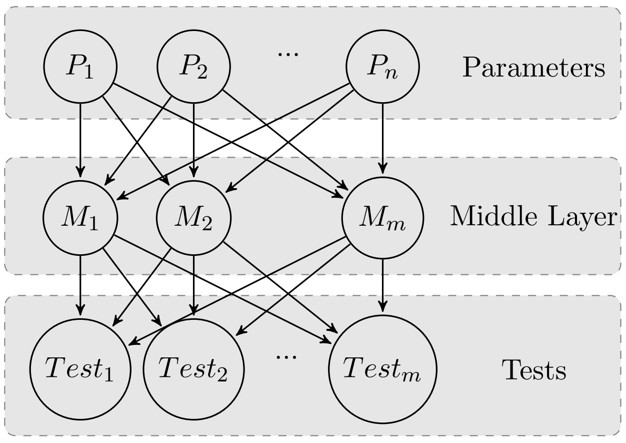


Figure 5.2: The network can be divided into three layers: parameter (top), middle layer, and tests (bottom) layers.

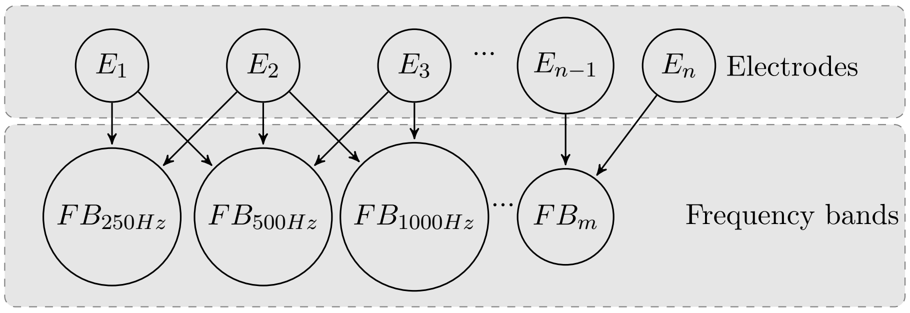


Figure 5.3: The network can be seen as a set of electrodes (or electrode groups) that affect a set of frequency bands

algorithms, we decided to implement Likelihood Weighting ([14, 26]), for its simplicity and good performance. The main advantage of using an stochastic inference algorithm such as Likelihood Weighting is that both the computation and memory cost grow in a linear fashion with the number of nodes whereas in exact inference it grows exponentially. The main drawback of Likelihood Weighting is its accuracy decreases when doing inference with extremely unlikely evidence, but it serves us well as the evidence will be mostly in nodes with no parents and no extreme probabilities.

5.2.5 Object Oriented Networks

As a quick look to the structure suggests, the model contains repetitive structures that are almost identical. The nodes could be grouped to form components or put it another way, the model could be defined as a set of components that are inter-connected. After analyzing the model with this perspective, we applied the Object-Oriented framework described in Section 3.2. Finally, the model has been re-structured as a set of objects of different classes: electrodes, frequency bands and tests.

5.2.6 Integration with OptiFox

OptiFox is also the name of the application developed internally by Otoconsult. It provides the user with a Graphical User Interface to introduce the data of the patient and present the recommendation from the system. Therefore we needed to find a way to communicate OptiFox, which is implemented using Microsoft's .NET Framework and OpenMarkov. For that purpose, a webservice was implemented offering the necessary functionality to retrieve a list of tuning recommendations given a network with a set of evidence data (in this case, the absolute values of parameters and the results of tests). Then the developers of Otoconsult implemented the necessary functionality to keep the webservice available, call it from within the application and show the results through the GUI in a user-friendly manner.

5.3 Evaluation and Results

The current state of the network just covers the lower frequencies (up to 1000Hz), which is around a fourth part of the spectrum the network is supposed to cover. Various (around 10) tests have been done with data taken from real patients with issues in the lower frequencies. In these tests the outcome of our model is evaluated comparing it against the outcome of FOX, the previous rule-based system and by the expert.

After some fine tuning of the probabilities by the expert, the model has reached the point where in most cases it agrees with the outcome of FOX and in some cases it does things differently. Also, in most cases the outcome has been the expected by the expert but in all cases the outcome of the model has been described by the expert as being "smart" and "intelligent".

So far, our prototype has been tried on only one CI user: a lady whose CI device had been fitted by expert audiologists supported by FOX was showing a poor performance at verbal audiometries. On July the 31st 2012, when audiologists programmed her CI following the recommendations of our probabilistic model, her ability to recognize speech increased to the range of people with normal hearing ability. Obviously, it is not possible to draw conclusions from one particular case, but this small success is encouraging given that there is still a lot of room for improvement in our model.

Chapter 6

Conclusions

6.1 Main Contributions

6.1.1 Interactive Learning

We have designed and implemented an interactive learning approach for learning Bayesian networks from databases, which may be very useful for the experts in different application domains, as well as for researchers and students in the field of PGMs.

We have presented a case study based on a well-known model, the ALARM network [5], frquently used in the literature on learning BNs. This study has shown that even very rudimentary causal knowledge about the domain may lead to a significant improvement of the network built interactively with the facilities we have developed.

Its main shortcoming is that the computation cost grows exponentially with the number of variables, which makes the learning of big networks unfeasible or a nuisance at best.

The Interactive Learning module is described on the paper "Interactive learning of Bayesian Networks with OpenMarkov" that will be presented in the upcoming Sixth European Workshop on Probabilistic Graphical Models.

6.1.2 Object Oriented Networks

We have studied the existing literature on Object Oriented Bayesian Networks as well as other frameworks that combine probabilistic models with relational models or even first order logic. Based on this study of the state of the art, we have come up with our own framework picking what we believe is the best of each of the studied proposals.

We have also designed an implemented in OpenMarkov the necessary functionality to build and edit Object Oriented Networks on a user-friendly way, making it one of the few tools available that support it.

6.1.3 Tuning Networks

We have presented a new type of probabilistic network focused on solving the problem of tuning systems with tunable parameters for their optimal performance. The main feature of this type of network is the use of a new canonical model that we call the tuning model. We have defined mathematically this model, analyzed its properties and developed efficient methods to integrate this model into standard inference algorithms, both exact and stochastic.

We have developed an algorithm for finding near-optimal interventions using tuning networks. We have implemented in OpenMarkov the tuning model and the algorithms for doing inference on tuning networks .

6.1.4 Application to Cochlear Implant programming

In the context of the European project Opti-FOX, we have built a tuning network for the programming of Cochlear Implant devices in collaboration with experts of Belgium and the Netherlands. In fact, the development of tuning networks and our framework for OONs has been motivated by the needs encountered in this project.

This model has been tested on a set of cases taken from a database of real CI users. The recommendations of our prototype have not been applied to these patients but the expert leading this project has valued them very positively. In most cases he agreed with these recommendations while in the rest he has considered the recommendations as different from what he would apply but nonetheless “intelligent”, “smart” and “worth trying”. In fact, the only CI user that has been programmed following our model’s recommendations, which differed from audiologists’ previous interventions, has experienced a noticeable increase in her speech understanding capability.

The advantages of our model with respect to the FOX rule based system are that our model is capable of complex reasoning whereas FOX only concatenates rules, that FOX is deterministic while our model handles uncertainty and our model will be fine-tuned by learning from data. Anyway, FOX is still a more mature project and includes features that our model still does not, such as the ability to specify the quantity by which the value of a parameter should be changed.

6.2 Future Work

6.2.1 Interactive Learning

The main lines for future development would be to borrow some ideas from the work of [29] and [8], such as representing graphically the strength of the correlation between variables and having richer types of constraints. It would be also useful to show an absolute quality measure of the net rather than the incremental one we currently have, given by the complexity of the network and the distance between the probability distribution of the network and that of the data. This quality measure could be used to compare the resulting nets of the interactive and non-interactive learning processes.

Another research line would be to adapt our approach to learning Bayesian classifiers, a somewhat different problem, as the objective is not to build the network that better represents the probability distribution of the data, but the network that better classifies new cases.

6.2.2 Object Oriented Networks

Object Oriented Networks still have a long way to go. Currently they have improved the way of building large PGMs, based on the reuse of building blocks that represent objects but the Object Oriented Programming paradigm is much more than that. Inheritance is defined in Bangsø and Wuillemin ([4]), just as a mechanism where ”the subtype adds things to the type”. Other Object Oriented mechanisms such as polimorphism could be defined for

Object Oriented Networks. Besides, inference can be optimized in the same way learning is optimized, taking advantage of the object oriented aspect of these models, reusing partial results across all instances of the same class.

6.2.3 Tuning Networks

A possible line for future research is to explore the behavior of the tuning model when using different combination functions; thus, instead of the "democratic" function f_{tuning}, defined in Equation 4.1, which assigns the same weight to each parent, we might have a function in which each parent "votes" with a different weight. However, such a function would require more parameters, and the increase in the complexity of the model, instead of improving its accuracy, might be counterproductive.

Another line of research is to improve the integration of the tuning model with exact and/or stochastic algorithms, in order to improve their efficiency.

6.2.4 Application to Cochlear Implant programming

The most obvious next step in the project is to test the developed prototype on real patients with different types of poor hearing ability and debug it accordingly.

Something we have been working on is learning the conditional probabilities from a database. Given that our model contained a middle layer of unobserved or latent variables and that the records of the database are seldom complete (as usually patients are not submitted to all tests), the usual parametric learning algorithms can not be applied. Lauritzen applied the Expectation Maximization (EM) algorithm [9] to Bayesian Networks [23] and achieved good results with missing data, be it with latent variables or missing values in the records. Therefore we are implementing the EM algorithm to fine tune the CPTs elicited by the expert.

Another shortcoming of our model is it does not model the individual electrodes. Rather, the 22 electrodes are grouped into six bands, corresponding to different frequencies, and all the electrodes inside a certain band are treated as a unit: all their parameters are increased or decreased by the same amount. For this reason, it would be advisable to build a refined model in which every electrode, with its parameters, is represented individually. That model would permit to decide which electrodes should be deactivated in some situations and how the bands should be reassigned between the remaining electrodes, and then to decide how the parameters of each electrode should be adjusted. This increase in the degree of detail is expected to lead to more accurate fittings.

Another aspect with room for improvement is the granularity of the variables Even if most parameters of CIs are numeric, because of the complexity of the domain and the difficulty of dealing with continuous variables in PGMs, we decided to discretize every variable into three levels. However this simplification prevents our model from accurately predicting the effect of small changes in the values of the parameters. Given that our goal is to improve the accuracy and the precision of the current model, it is clear that the new model should represent each parameter as a continuous variable, but this will require the development of new algorithms for inference and for learning the parameters of the network from data.

Besides, programming a cochlear implant usually involves several sessions, and the history of each patient provides useful information that should be taken into account. For example, if an increase in a parameter caused discomfort in the past, the programmer should not raise it to that level again. Unfortunately, the current model only considers the values of the parameters in the present. Additionally, even if the current fitting is close to optimal, it may be worthy to explore other configurations to further improve the user's performance. In the field of reinforcement learning, thoroughly investigated as a branch of artificial intelligence, this problem is described as a trade-off between exploration and exploitation. Nowadays, one of the most successful models for reinforcement learning are partially-observable Markov decision processes (POMDPs), which are an extension of Bayesian networks and influence diagrams in which time is represented explicitly. Turning a model as big as ours into a POMDP is a tremendous scientific challenge it is worth facing, since the power of such a model would exceed by far the cognitive capabilities of any human expert, and consequently its capability to determine the optimal sequence of tests and parameter adjustments would outperform significantly the expertise of the best human programmers of cochlear implants.

Bibliography

[1] M. Arias and F. J. Díez. Carmen: An open source project for probabilistic graphical models. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, pages 25–32, Hirtshals, Denmark, 2008.

[2] M. Arias, F. J. Díez, and M. P. Palacios. ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain, 2011.

[3] G. Baio, F. Pammolli, V. Baldo, and R. Trivello. Object oriented influence diagram for cost-effectiveness analysis of influenza vaccination in the Italian elderly population. *Expert Review of Pharmacoeconomics & Outcomes Research*, 6:293–301, 2006.

[4] O. Bangsø and P. H. Wuillemin. Top-down construction and repetetive structures representation in Bayesian networks. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS-2000)*, pages 282–286, Orlando, FL, 2000.

[5] I. A. Beinlich, H. J. Suermondt, R. M. Chávez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on AI and Medicine*, pages 247–256, London, 1989. Springer-Verlag, Berlin.

[6] R. R. Bouckaert. Bayesian networks in Weka. Technical Report 14/2004, Computer Science Department, University of Waikato, New Zealand, 2004.

[7] G. F. Cooper and E. Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence (UAI'91)*, pages 86–94, Los Angeles, CA, 1991. Morgan Kaufmann, San Mateo, CA.

[8] L. M. de Campos and J. G. Castellano. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45:233–254, 2007.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[10] F. J. Díez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain, 2006.

[11] M. J. Druzdzel and F. J. Díez. Combining knowledge from different sources in probabilistic models. *Journal of Machine Learning Research*, 4:295–316, 2003.

[12] The Elvira Consortium. Elvira: An environment for creating and using probabilistic graphical models. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain, 2002.

[13] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conferences on Artificial Intelligence*, pages 1300–1309, 1999.

[14] R. Fung and K. C. Chang. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In M. Henrion, R. D. Shachter, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 209–219, Amsterdam, The Netherlands, 1990. Elsevier Science Publishers.

[15] C. Glymour and G. F. Cooper. *Computation, Causation and Discovery.* The MIT Press, Cambridge, Massachusetts, 1999.

[16] P. J. Govaerts, B. Vaerenberg, G. De Ceulaer, K. Daemers, C. De Beukelaer, and K. Schauwers. Development of a software tool using deterministic logic for the optimization of cochlear implant processor programming. *Otology & Neurology*, 31:908–918, 2010.

[17] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA, 1984.

[18] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference in Artificial Intelligence (UAI-97)*, pages 302–313, San Francisco, CA, 1997. Morgan Kaufmann.

[19] C. Lacave and F. J. Díez. A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127, 2002.

[20] C. Lacave, M. Luque, and F. J. Díez. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965, 2007.

[21] C. Lacave, A. Oniśko, and F. J. Díez. Use of Elvira's explanation facilities for debugging probabilistic expert systems. *Knowledge-Based Systems*, 19:730–738, 2006.

[22] H. Langseth and O. Bangsø. Parameter learning in object-oriented bayesian networks. *Ann. Math. Artif. Intell.*, 32(1-4):221–243, 2001.

[23] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Comput. Stat. Data Anal.*, 19(2):191–201, February 1995.

[24] S. M. Mahoney and K. B. Laskey. Network engineering for complex belief networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI'96)*, pages 389–396, Portland, OR, 1996. Morgan Kaufmann, San Francisco, CA.

[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[26] R. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In M. Henrion, R. D. Shachter, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 221–231. Elsevier Science Publishers, Amsterdam, The Netherlands, 1990.

[27] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.

[28] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search.* The MIT Press, Cambridge, Massachusetts, second edition, 2000.

[29] L. E. Sucar and M. Martínez-Arroyo. Interactive structural learning of bayesian networks. *Expert Systems with Applications*, 15:325–332, 1998.

[30] Z. Szlavik, B. Vaerenberg, W. Kowalczyk, and P. Govaerts. Opti-fox: towards the automatic tuning of cochlear implants. In *Proceedings of the 20th Belgian Dutch Conference on Machine Learning*, pages 79–80, 2011.

[31] R. L. Teach and E. H. Shortliffe. An analysis of physician's attitudes. In B. G. Buchanan and E. H. Shortliffe, editors, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 34, pages 635–652. Addison-Wesley, Reading, MA, 1984.

[32] B. Vaerenberg, P. J. Govaerts, G. De Ceulaer, K. Daemers, and K. Schauwers. Experiences of the use of fox, an intelligent agent, for programming cochlear implant sound processors in new users. *International Journal of Audiology*, 50:50–58, 2011.