INRIA
RHÔNE-ALPES

Survey: Probabilistic Methodology and Techniques for Artefact Conception and Development

LAPLACE RESEARCH GROUP[1]

Projet SHARP

Thème 3 : Interaction homme-machine, images, données, connaissances

Pierre.Bessiere@imag.fr

November 2002

Abstract: The purpose of this paper is to make a state of the art on probabilistic methodology and techniques for artefact conception and development. It is the 8th deliverable of the BIBA (Bayesian Inspired Brain and Artefact) project. We first present the *incompletness* problem as the central difficulty that both living creatures and artefacts have to face: how can they perceive, infer, decide and act efficiently with incomplete and uncertain knowledge?. We then introduce a generic probabilistic formalism called *Bayesian Programming*. This formalism is then used to review the main probabilistic methodology and techniques. This review is organized in 3 parts: first the probabilistic models from Bayesian networks to Kalman filters and from sensor fusion to CAD systems, second the inference techniques and finally the learning and model acquisition and comparison methodologies. We conclude with the perspectives of the BIBA project as they rise from this state of the art.

1. PIERRE BESSIÈRE, JUAN-MANUEL AHUACTZIN, OLIVIER AYCARD, DAVID BELLOT, FRANCIS COLAS, CHRISTOPHE COUÉ, JULIEN DIARD, RUBEN GARCIA, CARLA KOIKE, OLIVIER LEBELTEL, RONAN LEHY, OLIVIER MALRAIT, EMMANUEL MAZER, KAMEL MEKHNACHA, CÉDRIC PRADALIER, ANNE SPALANZANI

1. INCOMPLETENESS AND UNCERTAINTY

We think that over the next decade, probabilistic reasoning will provide a new paradigm for understanding neural mechanisms and the strategies of animal behaviour at a theoretical level, and will raise the performance of engineering artefacts to a point where they are no longer easily outperformed by the biological examples they are imitating.

The BIBA project has been motivated by this conviction and aims to advance in this direction.

We assume that both living creatures and artefacts have to face the same fundamental difficulty: *incompleteness* (and its direct consequence *uncertainty)*. Any model of a real phenomenon is *incomplete*: there are always some hidden variables, not taken into account in the model, that influence the phenomenon. The effect of these hidden variables is that the model and the phenomenon never behave exactly alike. Both living organisms and robotic systems must face this central difficulty: how to use an *incomplete* model of their environment to perceive, infer, decide and act efficiently?

Rational reasoning with incomplete information is quite a challenge for artificial systems. The purpose of probabilistic inference and learning is precisely to tackle this problem with a well-established formal theory. During the past years a lot of progress has been made in this field both from the theoretical and applied point of view. The purpose of this paper is to give an overview of these works and especially to try a synthetic presentation using a generic formalism named Bayesian Programming (BP) to present all of them. It is not an impartial presentation of this field. Indeed, we present our own subjective "subjectivist" point of view and we think that is why it may be interesting.

The paper, after discussing further the *incompleteness* and *uncertainty* problems and how probabilistic reasoning helps deal with them, is organised in 4 main parts. The first part is a short and formal presentation of BP. The second part is a presentation of the main probabilistic models found in the literature. The third part describes the principal techniques and algorithms for probabilistic inference. Finally, the fourth deals with the learning aspects.

1.1 Incompleteness and uncertainty in robotics

The dominant paradigm in robotics may be caricatured by Figure 1.

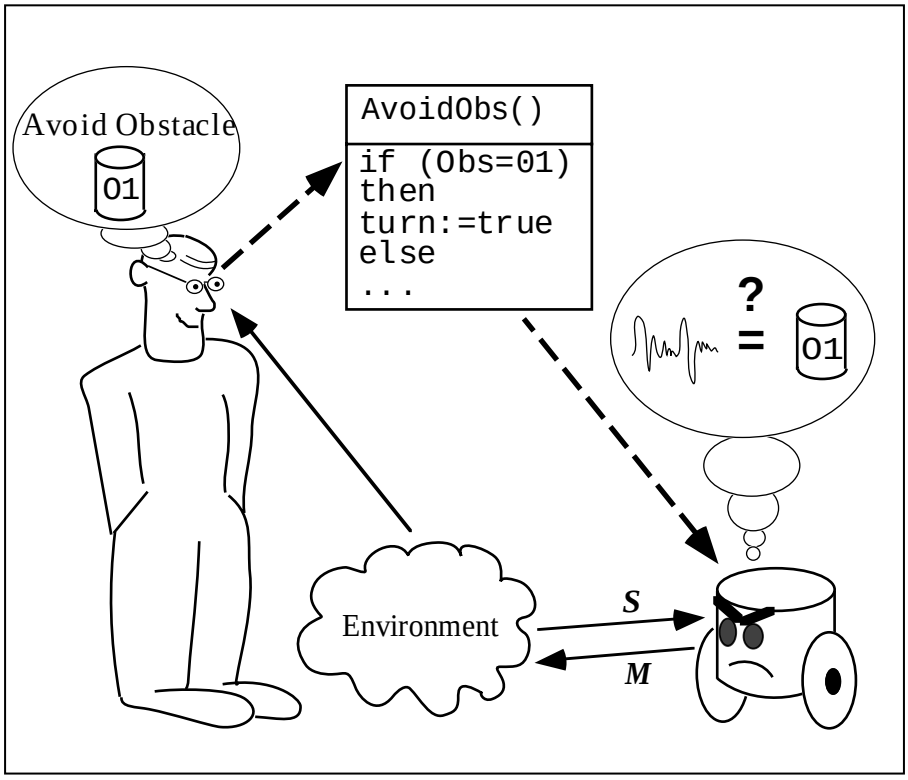

Figure 1: The symbolic approach in robotics.

The programmer of the robot has an abstract conception of its environment. He or she can describe the environment in geometrical terms because the shape of objects and the map of the world can be speci-

fied. He or she may describe the environment in analytical terms because the laws of physics that govern this world are known. The environment may also be described in symbolic terms because both the objects and their characteristics can be named.

The programmer uses this abstract representation to program the robot. The programs use these geometric, analytic and symbolic notions. In a way, the programmer imposes on the robot his or her own abstract conception of the environment.

The difficulties of this approach appear when the robot needs to link these abstract concepts with the raw signals it obtains from its sensors and sends to its actuators.

The central origin of these difficulties is the irreducible incompleteness of the models. Indeed, there are always some hidden variables, not taken into account in the model, that influence the phenomenon. The effect of these hidden variables is that the model and the phenomenon never behave exactly the same. The hidden variables prevent the robot from relating the abstract concepts and the raw sensory-motor data reliably. The sensory-motor data are then said to be «noisy» or even «aberrant». An odd reversal of causality occurs that seem to consider that the mathematical model is exact and that the physical world has some unknown flaws.

Controlling the environment is the usual answer to these difficulties. The programmer of the robot looks for the causes of «noises» and modifies either the robot or the environment to suppress these «flaws». The environment is modified until it corresponds to its mathematical model. This approach is both legitimate and efficient from an engineering point of view. A precise control of both the environment and the tasks ensures that industrial robots work properly.

However, compelling the environment may not be possible when the robot must act in an environment not specifically designed for it. In that case, completely different solutions must be devised.

1.2. Probabilistic approaches in robotics

The purpose of this paper is to present the probabilistic methodologies and techniques as a possible solution to the incompleteness and uncertainty difficulties.

Figure 2 introduces the principle of this approach.

The fundamental notion is to place side by side the programmer's conception of the task (the preliminary knowledge) and the experimental data to obtain probability distributions. These distributions can be used as programming resources.

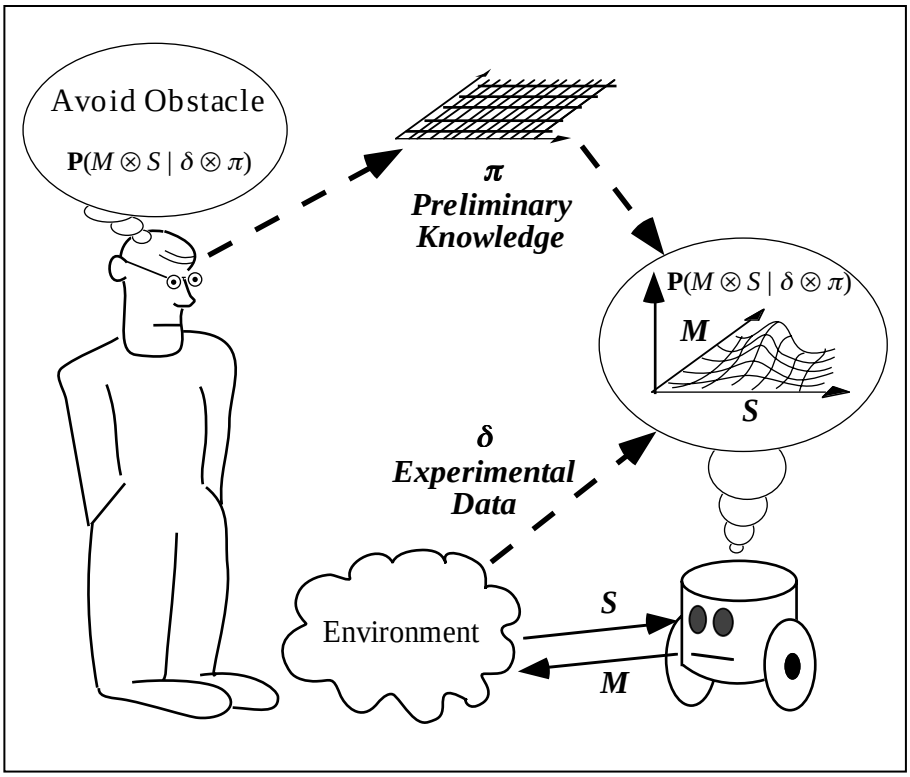

Figure 2: The probabilistic approaches in robotics.

The preliminary knowledge gives some hints to the robot about what it may expect to observe. The preliminary knowledge is not a fixed and rigid model purporting completeness. Rather, it is a gauge, with free parameters, waiting to be molded by the experimental data. Learning is the mean of setting these parameters. The pair made of a preliminary knowledge and set of experimental data is a *description*. The descriptions result from both the views of the programmer and the physical specificities of each robot and environment. Even the influence of the hidden variables is taken into account and quantified; the more important their effects, the more noisy the data, therefore the more uncertain the resulting probability distributions will be.

The probabilistic approaches to robotics usually need two steps as presented Figure 3.

The first step transforms the irreducible incompleteness into uncertainty. Starting from the prelimi-

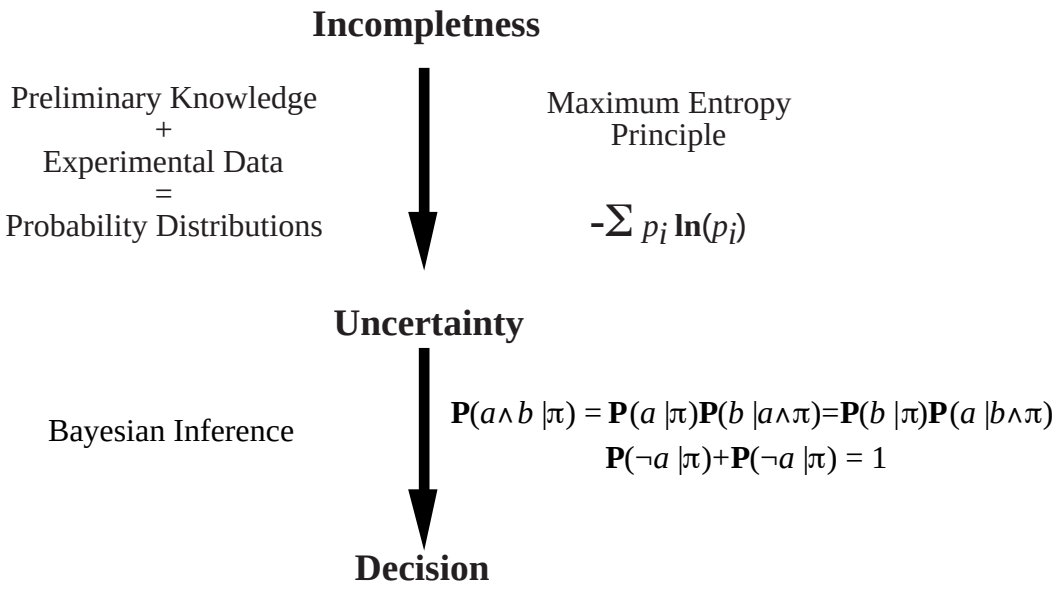

Figure 3: Theoretical foundation.

nary knowledge and the experimental data, learning builds probability distributions. The preliminary knowledge, even imperfect and incomplete, is relevant and provides interesting hints about the observed phenomenon. The more accurate and pertinent this preliminary knowledge is, the less uncertain and the more informational the learned distributions are.

The second step consists of reasoning with the probability distributions obtained by the first step. To do so, we only require the two basic rules of Bayesian inference. These two rules are to Bayesian inference what the resolution principle is to logical reasoning (see Robinson, 1965; Robinson, 1979; Robinson & Sibert, 1983a; Robinson & Sibert, 1983b). These inferences may be as complex and subtle as those achieved with logical inference tools.

Development on these arguments may be found in 2 papers by Bessière (Bessière et al., 1998a & Bessière et al., 1998b).

2. A GENERIC FORMALISM: THE BAYESIAN PROGRAMMING

In this section, we introduce the concepts, postulates, definitions, notations and rules that are necessary to define a Bayesian program. The Bayesian Program (BP) formalism will then be used in the sequel of this paper to present, compare and comment on a number of probabilistic models.

It may appear very basic and obvious but one of the goal of this survey is to demonstrate that such simple rules and formalism are sufficient to present a unifying framework for most of the probabilistic approaches found in the literature.

2.1. Definition and notation

2.1.1. Proposition

The first concept we will use is the usual notion of *logical proposition*. Propositions will be denoted by lowercase names. Propositions may be composed to obtain new propositions using the usual logical operators: $a \wedge b$ denotes the conjunction of propositions a and b, $a \vee b$ their disjunction and $\neg a$ the negation of proposition a.

2.1.2. Variable

The notion of *discrete variable* is the second concept we require. Variables will be denoted by names starting by an uppercase letter.

By definition, a *discrete variable* X is a set of logical propositions x_i such that these propositions are mutually exclusive (for all i, j with $i \neq j$, $x_i \wedge y_j$ is false) and exhaustive (at least one of the propositions x_i is true). x_i stands for «variable X takes its i^{th} value». $\lfloor X \rfloor$ denotes the cardinal of the set X (the number of propositions x_i).

The conjunction of two variables X and Y, denoted $X \wedge Y$, is defined as the set of $\lfloor X \rfloor \times \lfloor Y \rfloor$ propositions $x_i \wedge y_j$. $X \wedge Y$ is a set of mutually exclusive and exhaustive logical propositions. As such, it is a new variable[2]. Of course, the conjunction of N variables is also a variable and, as such, it may be renamed at any time and considered as a unique variable in the sequel.

2.1.3. Probability

To be able to deal with uncertainties, we will attach probabilities to propositions.

We consider that, to assign a probability to a proposition a, it is necessary to have at least some *preliminary knowledge*, summed up by a proposition π. Consequently, the probability of a proposition a is always conditioned, at least, by π. For each different π, $\mathbf{P}(. \mid \pi)$ is an application assigning to each proposition a a unique real value $\mathbf{P}(a \mid \pi)$ in the interval $[0, 1]$.

Of course, we will be interested in reasoning on the probabilities of the conjunctions, disjunctions and negations of propositions, denoted, respectively, by $\mathbf{P}(a \wedge b \mid \pi)$, $\mathbf{P}(a \vee b \mid \pi)$ and $\mathbf{P}(\neg a \mid \pi)$.

We will also be interested in the probability of proposition a conditioned by both the preliminary knowledge π and some other proposition b. This will be denoted $\mathbf{P}(a \mid b \wedge \pi)$.

For simplicity and clarity, we will also use probabilistic formulas with variables appearing instead of propositions. By convention, each time a variable X appears in a probabilistic formula $\Phi(X)$, it should be understood as $\forall x_i \in X, \Phi(x_i)$. For instance, given three variables X, Y and Z, $\mathbf{P}(X \wedge Y \mid Z \wedge \pi) = \mathbf{P}(X \mid \pi)$ stands for:

$$\begin{array}{c} \forall x_i \in X, \forall y_j \in Y, \forall z_k \in Z \\ \mathbf{P}(x_i \wedge y_j \mid z_k \wedge \pi) = \mathbf{P}(x_i \mid \pi) \end{array} \quad [1]$$

2.2. Inference postulates and rules

This section presents the inference postulates and rules necessary to carry out probabilistic reasoning.

2.2.1. Conjunction and normalization postulates for propositions

Probabilistic reasoning needs only two basic rules:

1 - The *conjunction rule*, which gives the probability of a conjunction of propositions.

2. By contrast, the disjunction of two variables, defined as the set of propositions $x_i \vee y_j$, is not a variable. These propositions are not mutually exclusive.

$$\begin{aligned}\mathbf{P}(a \wedge b \mid \pi) &= \mathbf{P}(a \mid \pi) \times \mathbf{P}(b \mid a \wedge \pi) \\ &= \mathbf{P}(b \mid \pi) \times \mathbf{P}(a \mid b \wedge \pi)\end{aligned} \quad [2]$$

2 - The *normalization rule*, which states that the sum of the probabilities of a and $\neg a$ is one.

$$\mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \mid \pi) = 1 \quad [3]$$

For the purpose of this paper, we take these two rules as postulates[3].

As in logic, where the resolution principle (Robinson, 1965; Robinson, 1979) is sufficient to solve any inference problem, in discrete probabilities, these two rules ([2], [3]) are sufficient for any computation. Indeed, we may derive all the other necessary inference rules from those two, especially the rules concerning variables.

2.2.2. Disjunction rule for propositions:

$$\begin{aligned}&\mathbf{P}(a \vee b \mid \pi) \\ &= (\mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) - \mathbf{P}(a \wedge b \mid \pi))\end{aligned} \quad [4]$$

2.2.3. Conjunction, normalization and maginalization rules for variables

Conjunction (Bayes) rule

$$\begin{aligned}\mathbf{P}(X \wedge Y \mid \pi) &= \mathbf{P}(X \mid \pi) \times \mathbf{P}(Y \mid X \wedge \pi) \\ &= \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \wedge \pi)\end{aligned} \quad [5]$$

Normalization rule

$$\sum_X \mathbf{P}(X \mid \pi) = 1 \quad [6]$$

Marginalization rule for variables

$$\sum_X \mathbf{P}(X \wedge Y \mid \pi) = \mathbf{P}(Y \mid \pi) \quad [7]$$

2.3. Bayesian Programs

Using these very simple postulates and rules it is possible to define a generic formalism to specify probabilistic models: Bayesian Programming (BP). This formalism will be used in the sequel of this paper to describe all the presented models.

The elements of a Bayesian Program are presented Figure 4:

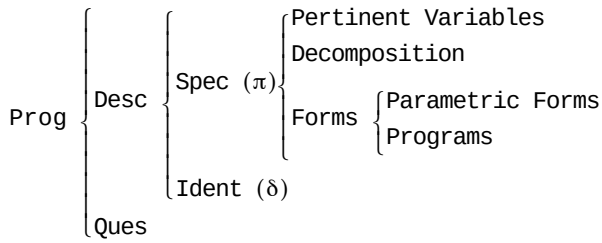

Figure 4: Structure of a bayesian program

- A program is constructed from a description, which constitutes a knowledge base (declarative part), and a question, which restitutes via inference some of this knowledge (procedural part).
- A description is constructed in 2 phases: a specification phase where the programmer expresses its preliminary knowledge and an indentification (or learning) phase where the experimental data are taken into account.
- Preliminary knowledge is constructed from a set of pertinent variables, a decomposition of the joint distribution into a product of simpler terms, and a set of forms, one for each term.
- Forms are either parametric forms or questions to other Bayesian programs.

3. Theoretical justifications of probabilistic inference and maximum entropy are numerous. The entropy concentration theorems (Jaynes, 1982; Robert, 1990) are among the more rigorous, Cox theorem (Cox, 1961) being the most well known, although it has been partially disputed recently by Halpern (Halpern, 1999a; Halpern, 1999b).

2.3.1. Description

The purpose of a description is to specify an effective method to compute a joint distribution on a set of variables $\{X^1, X^2, ..., X^n\}$, given a set of experimental data δ and a preliminary knowledge π. This joint distribution is denoted as: $\mathbf{P}(X^1 \wedge X^2 \wedge ... \wedge X^n \mid \delta \wedge \pi)$.

2.3.2. Preliminary Knowledge

To specify a preliminary knowledge the programmer must:

1 Define the set of relevant variables $\{X^1, X^2, ..., X^n\}$ on which the joint distribution is defined.

2 Decompose the joint distribution:

Given a partition of $\{X^1, X^2, ..., X^n\}$ into k subsets, we define k variables $L^1, ..., L^k$, each being the conjunction of the variables in each of these subsets.

The conjunction rules [5] leads to:

$$\mathbf{P}(X^1 \wedge X^2 \wedge ... \wedge X^n \mid \delta \wedge \pi) = \mathbf{P}(L^1 \mid \delta \wedge \pi) \times \mathbf{P}(L^2 \mid L^1 \wedge \delta \wedge \pi) \times ... \times \mathbf{P}(L^k \mid L^{k-1} \wedge ... \wedge L^2 \wedge L^1 \wedge \delta \wedge \pi) \quad [8]$$

Conditional independence hypotheses then allow further simplifications. A conditional independence hypothesis for variable L^i is defined by choosing some variables X^j among the variables appearing in conjunction $L^{i-1} \wedge ... \wedge L^2 \wedge L^1$, calling R^i the conjunction of these chosen variables and setting:

$$\mathbf{P}(L^i \mid L^{i-1} \wedge ... \wedge L^2 \wedge L^1 \wedge \delta \wedge \pi) = \mathbf{P}(L^i \mid R^i \wedge \delta \wedge \pi) \quad [9]$$

We then obtain:

$$\mathbf{P}(X^1 \wedge X^2 \wedge ... \wedge X^n \mid \delta \wedge \pi) = \mathbf{P}(L^1 \mid \delta \wedge \pi) \times \mathbf{P}(L^2 \mid R^2 \wedge \delta \wedge \pi) \times \mathbf{P}(L^3 \mid R^3 \wedge \delta \wedge \pi) \times ... \times \mathbf{P}(L^k \mid R^k \wedge \delta \wedge \pi) \quad [10]$$

Such a simplification of the joint distribution as a product of simpler distributions is called a decomposition.

3 Define the forms:

Each distribution $\mathbf{P}(L^i \mid R^i \wedge \delta \wedge \pi)$ appearing in the product is then associated with either a parametric form (i.e., a function $f_\mu(L^i)$) or another Bayesian program. In general, μ is a vector of parameters that may depend on R^i or δ or both. Learning takes place when some of these parameters are computed using the data set δ.

2.3.3. Question

Given a description (i.e., $\mathbf{P}(X^1 \wedge X^2 \wedge ... \wedge X^n \mid \delta \wedge \pi)$), a question is obtained by partitioning $\{X^1, X^2, ..., X^n\}$ into three sets : the searched variables, the known variables and the unknown variables.

We define the variables *Searched*, *Known* and *Unknown* as the conjunction of the variables belonging to these sets. We define a question as the distribution:

$$\mathbf{P}(Searched \mid Known \wedge \delta \wedge \pi). \quad [11]$$

See section 4, “Bayesian Reasonning” on page 25, for all developments on how this question may be answered.

3. BAYESIAN PROGRAMMING AND MODELLING

The goal of this section is to present the main probabilistic models currently used for artefact conception and development.

We will systematically use the Bayesian programming formalism to present these models. This is a good manner to be precise and concise and will simplify their comparison.

We will mainly concentrate on the definition of these models: discussions about inference and computation will be postponed to section 4 and discussions about learning and identification will be postponed to section 5.

We chose to divide the different probabilistic models into 2 categories: the general purpose probabilistic models and the problem oriented probabilistic models.

In the first category, the modelling choices are made independently of any specific knowledge about the modeled phenomenon. Most of the time these choices are essentially made to keep with tractable inference. However, the technical simplifications of these models may be compatible with large classes of problems and consequetly may have numerous applications.

In the second category, on the contrary, the modelling choices and simplifications are decided according to some specific knowledge about the modeled phenomenon. These choices could eventually lead to very poor models from a computational point of view. However, most of the time, problem dependent knowledge like, for instance, conditional independence between variables, leads to very important and effective simplifications and computational improvements.

3.1. General Purpose Probabilistic Models

3.1.1. Graphical Models and Bayesian Networks

Bayesian Networks

Bayesian networks, first introduced by Judea Pearl (Pearl, 1988), have emerged as a primary method for dealing with probabilistic and uncertain information. They are the result of the marriage between the theory of probabilities and the theory of graphs.

They are defined by the following Bayesian program:

$$\texttt{Prog}\begin{cases}\texttt{Desc}\begin{cases}\texttt{Spec}\begin{cases}\texttt{Pertinent Variables}\\ X_1, \ldots\ldots, X_N\\ \texttt{Decomposition}\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N) = \prod_{i=1}^{N} \mathbf{P}(X_i \mid R_i)\\ \texttt{Forms}\\ \textit{Any}\end{cases}\\ \texttt{Ident}\end{cases}\\ \texttt{Ques}\\ \mathbf{P}(X_i \mid \textit{Known})\end{cases} \qquad [12]$$

- The pertinent variables are not constrained and have no specific semantics.
- The decomposition, on the contrary, is specific: it is a product of distributions with one and only one variable X_i conditioned by a conjunction of other variables R_i called its "parents". An obvious bijection exists between joint probability distributions defined by such a decomposition and directed acyclic graphs: nodes are associated to variables, and oriented edges are associated to conditional dependencies. Using graphs in probabilistic models leads to an efficient way to define hypotheses over a set of variables, an economic representation of a joint probability distribution and, most importantly, an easy and efficient way to do probabilistic inference (see section 4.2.1).
- The parametric forms are not constrained but they are very often restricted to probability tables (this is, for instance, the case in the Bayesian networks commercial softwares such as Hugin[4] or Netica[5])
- Very efficient inference techniques have been developped to answer question $\mathbf{P}(X_i \mid \textit{Known})$,

4. http://www.hugin.com/
5. http://www.norsys.com/

however some difficulties appear for more general questions (see section 4.2.1).

Readings on Bayesian nets and graphical models should start by the following introductory textbooks: *Probabilistic reasoning in intelligent systems : Networks of plausible inference (*Pearl, 1988), *Graphical Models* (Lauritzen, 1996), *Learning in Graphical Models* (Jordan, 1998) and *Graphical Models for Machine Learning and Digital Communication* (Frey, 1998).

Dynamical Bayesian Networks

To deal with time and to model stochastic processes, the framework of Bayesian Networks has been extended to Dynamic Bayesian Networks (DBN) (see Dean & Kanazawa, 1989). Given a graph representing the structural knowledge at time t, supposing this structure to be time-invariant and time to be discrete, the resulting DBN is the repetition of the first structure from a start time to a final time. Each part at time t in the final graph is named a time slice.

They are defined by the following Bayesian program:

$$\texttt{Prog}\begin{cases}\texttt{Desc}\begin{cases}\texttt{Spec}\begin{cases}\texttt{Pertinent Variables}\\ X_1^0, \ldots\ldots, X_N^0, \ldots\ldots, X_1^T, \ldots\ldots, X_N^T\\ \texttt{Decomposition}\\ \mathbf{P}(X_1^0 \wedge \ldots\ldots \wedge X_N^T) = \mathbf{P}(X_1^0 \wedge \ldots\ldots \wedge X_N^0) \times \prod_{t=0}^{T}\prod_{i=1}^{N}\mathbf{P}(X_i^t \mid R_i^t)\\ \texttt{Forms}\\ \textit{Any}\end{cases}\\ \texttt{Ident}\end{cases}\\ \texttt{Ques}\\ \mathbf{P}(X_i^T \mid \textit{Known})\end{cases} \qquad [13]$$

- R_i^t is a conjunction of variables taken in the set $\{X_1^t, \ldots\ldots, X_{i-1}^t\} \cup \{X_1^{t-1}, \ldots\ldots, X_N^{t-1}\}$. It means that X_i^t depends only on its parents at time t ($\{X_1^t, \ldots\ldots, X_{i-1}^t\}$) as in a regular BN and on some variables from the previous time slice ($\{X_1^{t-1}, \ldots\ldots, X_N^{t-1}\}$).
- $\prod_{i=1}^{N}\mathbf{P}(X_i^t \mid R_i^t)$ defines a graph for a time slice and all time slices are identical when the time index t is changing.
- A DBN as a whole, "unrolled" over time, may be considered as a regular BN. Consequently the usual inference techniques applicable to BN are still valid for such "unrolled" DBNs (see section 4.2.1).

The best introduction, survey and starting point on DBNs is the Ph.D. thesis of K. Murphy entitled *Dynamic Bayesian Networks: Representation, Inference and Learning* (Murphy, 2002).

3.1.2. Recursive Bayesian Estimation, Hidden Markov Models, Kalman Filters and Particle Filters

Recursive Bayesian Estimation: Bayesian Filtering, Prediction and Smoothing

Recursive Bayesian Estimation is the generic denomination for a very largely applied class of numerous different probabilistic models of time series.

They are defined by the following Bayesian program:

$$
\texttt{Prog}\begin{cases}
\texttt{Desc}\begin{cases}
\texttt{Spec}\begin{cases}
\texttt{Pertinent Variables}\\
S^0, ..., S^N, O^0, ..., O^N\\
\texttt{Decomposition}\\
\mathbf{P}(S^0 \wedge ..... \wedge S^N \wedge O^0 \wedge ..... \wedge O^N) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{N} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\
\texttt{Forms}\\
\mathbf{P}(S^0)\\
\mathbf{P}(S^i \mid S^{i-1})\\
\mathbf{P}(O^i \mid S^i)
\end{cases}\\
\texttt{Ident}
\end{cases}\\
\texttt{Ques}\\
\mathbf{P}(S^{t+k} \mid O^0 \wedge ..... \wedge O^t)\\
(k = 0) \equiv Filtering\\
(k > 0) \equiv Prediction\\
(k < 0) \equiv Smoothing
\end{cases}
\qquad [14]
$$

- Variables $S^0, ..., S^N$ are a time series of "state" variables considered on a time horizon ranging from 0 to N. Variables $O^0, ..., O^N$ are a time series of "observation" variables on the same horizon.
- The decomposition is based:
  - on $\mathbf{P}(S^i \mid S^{i-1})$, called the "system model" or "transition model", which formalized the transition model from state at time i-1 to state at time i,
  - on $\mathbf{P}(O^i \mid S^i)$, called the "observation model", which expresses what can be observed at time i when the system is in state S^i,
  - and on a prior $\mathbf{P}(S^0)$ over states at time 0.
- The question usually asked to these models is $\mathbf{P}(S^{t+k} \mid O^0 \wedge ..... \wedge O^t)$: what is the probability distribution for state at time t+k knowing the observations from instant 0 to t? The most common case is Bayesian Filtering where $k = 0$, which means that you search for the present state knowing the past observations. However it is also possible to do "prediction" $(k > 0)$ where one tries to extrapolate future state from passed observations, or to do "smoothing" $(k < 0)$ where one tries to recover a past state from observations made either before or after that instant. However, some more complicated questions may also be asked (see HMM further on).

Bayesian Filters $(k = 0)$ have a very interesting recursive property which contributes largely to their interest. $\mathbf{P}(S^t \mid O^0 \wedge ..... \wedge O^t)$ may be simply computed from $\mathbf{P}(S^{t-1} \mid O^0 \wedge ..... \wedge O^{t-1})$ with the following formula:

$$
\mathbf{P}(S^t \mid O^0 \wedge ..... \wedge O^t) = \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1}) \times \mathbf{P}(S^{t-1} \mid O^0 \wedge ..... \wedge O^{t-1})] \qquad [15]
$$

We give this derivation as an example of application of the rules presented in section 2.

$$
\begin{aligned}
&\mathbf{P}(S^t \mid O^0 \wedge \ldots\ldots \wedge O^t)\\
&= \sum_{S^0,\ldots\ldots,S^{t-1}} \left[\sum_{\substack{O^{t+1},\ldots\ldots,O^N\\ S^{t+1},\ldots\ldots,S^N}} \frac{\mathbf{P}(S^0 \wedge \ldots\ldots \wedge S^N \wedge O^0 \wedge \ldots\ldots \wedge O^N)}{\mathbf{P}(O^0 \wedge \ldots\ldots \wedge O^t)} \right]\\
&= \sum_{S^0,\ldots\ldots,S^{t-1}} \left[\sum_{\substack{O^{t+1},\ldots\ldots,O^N\\ S^{t+1},\ldots\ldots,S^N}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{N} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots\ldots \wedge O^t)} \right]\\
&= \sum_{S^0,\ldots\ldots,S^{t-1}} \left[\begin{array}{l} \dfrac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots\ldots \wedge O^t)}\\ \times \sum_{\substack{O^{t+1},\ldots\ldots,O^N\\ S^{t+1},\ldots\ldots,S^N}} \left[\prod_{j=t+1}^{N} [\mathbf{P}(S^j \mid S^{j-1}) \times \mathbf{P}(O^j \mid S^j)] \right] \end{array} \right]\\
&= \sum_{S^0,\ldots\ldots,S^{t-1}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots\ldots \wedge O^t)}\\
&= \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} \left[\mathbf{P}(S^t \mid S^{t-1}) \times \sum_{S^0,\ldots\ldots,S^{t-2}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t-1} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots\ldots \wedge O^{t-1})} \right]\\
&= \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1}) \times \mathbf{P}(S^{t-1} \mid O^0 \wedge \ldots\ldots \wedge O^{t-1})]
\end{aligned}
\qquad [16]
$$

Prediction and smoothing do not lead to such nice simplifications and suppose to make large sums that represent a huge computational burden.

Hidden Markov Models

The Hidden Markov Models (HMM) are a very popular specialization of Bayesian Filters.

They are defined by the following Bayesian program:

$$
\texttt{Prog} \begin{cases}
\texttt{Desc} \begin{cases}
\texttt{Spec} \begin{cases}
\texttt{Pertinent Variables}\\
S^0, \ldots\ldots, S^t, O^0, \ldots\ldots, O^t\\
\texttt{Decomposition}\\
\mathbf{P}(S^0 \wedge \ldots\ldots \wedge S^t \wedge O^0 \wedge \ldots\ldots \wedge O^t) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\
\texttt{Forms}\\
\mathbf{P}(S^0) \equiv \mathit{Matrix}\\
\mathbf{P}(S^i \mid S^{i-1}) \equiv \mathit{Matrix}\\
\mathbf{P}(O^i \mid S^i) \equiv \mathit{Matrix}
\end{cases}\\
\texttt{Ident}
\end{cases}\\
\texttt{Ques}\\
\mathbf{P}(S^1 \wedge S^2 \wedge \ldots\ldots \wedge S^{t-1} \mid S^t \wedge O^0 \wedge \ldots\ldots \wedge O^t)
\end{cases}
\qquad [17]
$$

- Variables are supposed to be discrete.
- The transition model $\mathbf{P}(S^i \mid S^{i-1})$ and the observation models $\mathbf{P}(O^i \mid S^i)$ are both specified using probability matrices.
- The most popular question asked to HMMs is $\mathbf{P}(S^1 \wedge S^2 \wedge \ldots\ldots \wedge S^{t-1} \mid S^t \wedge O^0 \wedge \ldots\ldots \wedge O^t)$: what is the most probable series of states that leads to the present state knowing the past observa-

tions?

This particular question may be answered with a specific and very efficient algorithm called the "Viterbi algorithm" which will be presented in section 4.3.4.

A specific learning algorithm called the "Baum-Welch" algorithm has also been developped for HMMs (see section 5.2.3)

A nice start about HMM is Rabiner's tutorial (Rabiner, 1989).

Kalman Filters

The very well known Kalman Filters (Kalman, 1960) are another specialization of Bayesian Filters.

They are defined by the following Bayesian program:

$$
\text{Prog}\begin{cases}
\text{Desc}\begin{cases}
\text{Spec}\begin{cases}
\text{Pertinent Variables}\\
S^0, \ldots\ldots, S^t, O^0, \ldots\ldots, O^t\\
\text{Decomposition}\\
\mathbf{P}(S^0 \wedge \ldots\ldots \wedge S^t \wedge O^0 \wedge \ldots\ldots \wedge O^t) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\
\text{Forms}\\
\mathbf{P}(S^0) \equiv \mathbf{G}(S^0, \mu, \sigma)\\
\mathbf{P}(S^i \mid S^{i-1}) \equiv \mathbf{G}(S^i, A \bullet S^{i-1}, Q)\\
\mathbf{P}(O^i \mid S^i) \equiv \mathbf{G}(O^i, H \bullet S^i, R)
\end{cases}\\
\text{Ident}
\end{cases}\\
\text{Ques}\\
\mathbf{P}(S^t \mid O^0 \wedge \ldots\ldots \wedge O^t)
\end{cases}
\qquad [18]
$$

- Variables are continuous.
- The transition model $\mathbf{P}(S^i \mid S^{i-1})$ and the observation models $\mathbf{P}(O^i \mid S^i)$ are both specified using Gaussian laws with means that are linear functions of the conditioning variables.

Due to these hypotheses, and using the recursive formula [15], it is possible to analytically solve the inference problem to answer the usual $\mathbf{P}(S^t \mid O^0 \wedge \ldots\ldots \wedge O^t)$ question. This leads to an extremely efficient algorithm that explains the popularity of Kalman Filters and the number of their everyday applications.

When there is no obvious linear transition and observation models, it is still often possible, using a first order Taylor's expansion, to consider that these models are locally linear. This generalization is commonly called extended Kalman filters.

A nice tutorial by Welch and Bishop may be found on the Web (Welch & Bishop, 1997). For a more complete mathematical presentation one should refer to a report by Barker et al (Barker, Brown & Martin, 1994) but these are only 2 entries to a huge literature concerning the subject.

Particle Filters

The fashionable Particle Filters may also be seen as a specific implementation of Bayesian Filters.

The distribution $\mathbf{P}(S^{t-1} \mid O^0 \wedge \ldots\ldots \wedge O^{t-1})$ is approximated by a set of N particles having weights proportional to their probabilities. The recursive equation [15] is then used to inspire a dynamic process that produces an approximation of $\mathbf{P}(S^t \mid O^0 \wedge \ldots\ldots \wedge O^t)$. The principle of this dynamical process is that the particles are first moved according to the transition model $\mathbf{P}(S^t \mid S^{t-1})$, then their weights are updated according to the observation model $\mathbf{P}(O^t \mid S^t)$.

See Arulampalam's tutorial for a start (Arulampalam et al., 2001).

3.1.3. Mixture Models

Mixture models try to approximate a distribution on a set of variables $\{X_1, \ldots\ldots, X_N\}$ by adding up (mixing) a set of simple distributions.

The most popular mixture models are Gaussian mixtures where the component distributions are Gaussians. However, the component distributions may be of any nature as for instance logistic or Poisson distributions. In the sequel, for simplicity, we will take the case of Gaussian mixtures.

Such a mixture is defined as follows:

$$\text{Prog}\begin{cases}\text{Desc}\begin{cases}\text{Spec}\begin{cases}\text{Pertinent Variables}\\ X_1, \ldots\ldots, X_N\\ \text{Decomposition}\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N) = \sum_{i=1}^{M} [\alpha_i \times \mathbf{P_i}(X_1 \wedge \ldots\ldots \wedge X_N)]\\ \text{Forms}\\ \mathbf{P_i}(X_1 \wedge \ldots\ldots \wedge X_N) \equiv \mathbf{G}(X_1 \wedge \ldots\ldots \wedge X_N, \mu_i, \sigma_i)\end{cases}\\ \text{Ident}\end{cases}\\ \text{Ques}\\ \mathbf{P}(Searched \mid Known)\end{cases} \quad [19]$$

It should be noticed that this is not a Bayesian program. In particular, the decomposition does not have the right form $\mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N) = \mathbf{P}(L^1 \mid \delta \wedge \pi) \times \prod_{i=2}^{M} \mathbf{P}(L^i \mid R^i \wedge \delta \wedge \pi)$, as defined in equation [10].

It is, however, a very popular and convenient way to specify distributions $\mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N)$. Especially when the type of component distributions $\mathbf{P_i}(X_1 \wedge \ldots\ldots \wedge X_N)$ is chosen to insure nice and efficient analytical solutions to some of the inference problems.

Furthermore, it is possible to specify such mixtures as a correct Bayesian program by adding variables to the previous definition:

$$\text{Prog}\begin{cases}\text{Desc}\begin{cases}\text{Spec}\begin{cases}\text{Pertinent Variables}\\ X_1, \ldots\ldots, X_N, Mu_1, \ldots\ldots, Mu_M, Sigma_1, \ldots\ldots, Sigma_M, H\\ \text{Decomposition}\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \wedge Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M \wedge H)\\ = \mathbf{P}(Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M) \times \mathbf{P}(H)\\ \times \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M \wedge H)\\ \text{Forms}\\ \mathbf{P}(Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M) \equiv Uniform\\ \mathbf{P}(H) \equiv Table\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M \wedge [H = i])\\ = \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid \mu_i \wedge \sigma_i)\end{cases}\\ \text{Ident}\\ \mathbf{P}(H)\\ \mu_1, \ldots\ldots, \mu_M, \sigma_1, \ldots\ldots, \sigma_M\end{cases}\\ \text{Ques}\\ \mathbf{P}(Searched \mid Known \wedge \mu_1 \wedge \ldots\ldots \wedge \mu_M \wedge \sigma_1 \wedge \ldots\ldots \wedge \sigma_M)\\ = \sum_{H} \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \wedge Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M \wedge H)\\ = \sum_{i=1}^{M} [\mathbf{P}([H = i]) \times \mathbf{P}(Searched \mid Known \wedge \mu_i \wedge \sigma_i)]\end{cases} \quad [20]$$

- $\{Mu_1, \ldots\ldots, Mu_M, Sigma_1, \ldots\ldots, Sigma_M\}$ is a set of variables corresponding to the parameters of the M component distributions. (*i.e.* the M means and standard deviations in the Gaussian example).
- H is a discrete variable taking M values. H is used as a selection variable. Knowing the value of H, we suppose that the joint distribution is reduced to one of its component distribution:

$$\begin{aligned}&\mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid Mu_1 \wedge \ldots\ldots \wedge Mu_M \wedge Sigma_1 \wedge \ldots\ldots \wedge Sigma_M \wedge [H = i])\\ &= \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid \mu_i \wedge \sigma_i)\end{aligned} \quad [21]$$

- In these simple and common mixture models, H, the mixing variable is suppose to be indepen-

dent of the other variables. We will see in the sequel other mixing models where H may depend on some of the other variables (see the combination model in section 3.2.7). It is also the case in expert mixture models as decribed by Jordan (see Jordan & Jacobs, 1994 or Meila & Jordan, 1996)

- Identification is there a crucial step where the values of $\mathbf{P}(H)$ and the parameters $\mu_1, ..., \mu_M, \sigma_1, ..., \sigma_M$ of the component distributions are searched in order to have the best possible fit between the observed data and the joint distribution. This is usally done using the EM algorithm or some of its variants (see section 5.2).
- The question asked to the joint distribution are of the form $\mathbf{P}(Searched \mid Known \wedge \mu_1 \wedge ..... \wedge \mu_M \wedge \sigma_1 \wedge ..... \wedge \sigma_M)$ where $Searched$ and $Known$ are conjunctions of some of the $X_1, ..., X_N$. The parameters $\mu_1 \wedge ..... \wedge \mu_M \wedge \sigma_1 \wedge ..... \wedge \sigma_M$ of the component distributions are known but not the selection variable H, which stays always hidden. Consequently, solving the question supposes to sum on the possible value of H and we finally retrieve the usual mixture form:

$$\begin{aligned}&\mathbf{P}(Searched \mid Known \wedge \mu_1 \wedge ..... \wedge \mu_M \wedge \sigma_1 \wedge ..... \wedge \sigma_M)\\ &= \sum_{i=1}^{M} [\mathbf{P}([H = i]) \times \mathbf{P}(Searched \mid Known \wedge \mu_i \wedge \sigma_i)]\end{aligned} \quad [22]$$

A reference document on mixture models is McLachlan's book entitled *Finite Mixture Models (*McLachlan & Deep, 2000*).*

3.1.4. Maximum Entropy Approaches

Maximum entropy approaches play a very important role in physical applications. The late E.T. Jaynes, in his unfortunately unachieved book (Jaynes, 1995), gives a wonderful presentation of them as well as a fascinating apologia of the subjectivist epistemology of probabilities.

The Maximum entropy models may be described by the following Bayesian program:

$$\texttt{Prog} \begin{cases} \texttt{Desc} \begin{cases} \texttt{Spec} \begin{cases} \texttt{Pertinent Variables} \\ X_1, ..., X_N \\ \texttt{Decomposition} \\ \mathbf{P}(X_1 \wedge ..... \wedge X_N) \\ = \frac{1}{Z} \times e^{-\sum_{i=1}^{M} [\lambda_i \times \mathbf{f_i}(X_1 \wedge ..... \wedge X_N)]} \\ = \frac{1}{Z} \times \prod_{i=1}^{M} e^{-[\lambda_i \times \mathbf{f_i}(X_1 \wedge ..... \wedge X_N)]} \\ \texttt{Forms} \\ \mathbf{f}_1, ..., \mathbf{f}_M \end{cases} \\ \texttt{Ident} \\ \lambda_1 \wedge ..... \wedge \lambda_M \end{cases} \\ \texttt{Ques} \\ \mathbf{P}(Searched \mid Known) \end{cases} \quad [23]$$

- The variables $X_1, ..., X_N$ are not constrained.
- The decomposition is made of a product of exponential distributions $e^{-[\lambda_i \times \mathbf{f_i}(X_1 \wedge ..... \wedge X_N)]}$ where each $\mathbf{f_i}$ is called an *observable function*. An observable function may be any real function on the space defined by $X_1 \wedge ..... \wedge X_N$, such that its expectation may be computed:

$$\langle \mathbf{f_i}(X_1 \wedge ..... \wedge X_N) \rangle = \sum_{X_1 \wedge ..... \wedge X_N} [\mathbf{P}(X_1 \wedge ..... \wedge X_N) \times \mathbf{f_i}(X_1 \wedge ..... \wedge X_N)]. \quad [24]$$

- The constraints on the problem are usually expressed by M real values F_i called *level of constraint* which impose that $\langle \mathbf{f_i}(X_1 \wedge ..... \wedge X_N) \rangle = F_i$.
- The identification problem is then, kowing the level of constraint F_i, to find the *Lagrange mutipliers* λ_i which maximize the entropy of the distribution $\mathbf{P}(X_1 \wedge ..... \wedge X_N)$.

The maximun entropy approach is a very general and powerful way to represent probabilistic models and to explain what is going on when one wants to identify the parameters of a distribution, choose its

form or even compare models. It, however, often leads to computations which are intractable.

A nice introduction is of course Jaynes' book entitled *Probability theory - The logic of science (*Jaynes, 1995*)*. References on the subject are the books, edited after the regular MaxEnt conferences, which cover both theory and applications (see Levine & Tribus, 1979; Erickson & Smith, 1988a; Erickson & Smith, 1988b; Kapur & Kesavan, 1992; Smith & Grandy, 1985; Mohammad-Djafari & Demoment, 1992 & Mohammad-Djafari, 2000).

3.2. Problem Oriented Probabilistic Models

3.2.1. Sensor Fusion

Sensor fusion is a very common and crucial problem for both living systems and artefacts. The problem is as follows: given a phenomenon and some sensors, how to get information on the phenomenon by putting together the information of the different sensors?

The most common and simple Bayesian modelling for sensor fusion is the following:

$$\texttt{Prog}\begin{cases}\texttt{Desc}\begin{cases}\texttt{Spec}\begin{cases}\texttt{Pertinent Variables}\\ \Phi, S_1, ..., S_N\\ \texttt{Decomposition}\\ \mathbf{P}(\Phi \wedge S_1 \wedge ..... \wedge S_N) = \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi)\\ \texttt{Forms}\\ \textit{Any}\end{cases}\\ \texttt{Ident}\end{cases}\\ \texttt{Ques}\\ \mathbf{P}(\Phi \mid S_1 \wedge ..... \wedge S_N) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi)\\ \mathbf{P}(\textit{Searched} \mid \textit{Known})\end{cases} \qquad [25]$$

- Φ is the variable used to describe the phenomenon, when $\{S_1, ..., S_N\}$ are the variables encoding the readings of the sensors.
- The decomposition $\mathbf{P}(\Phi \wedge S_1 \wedge ..... \wedge S_N) = \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi)$ may seem peculiar as obviously the readings of the different sensors are not independent from one another. The exact meaning of this equation is that the phenomenon Φ is considered to be the main reason for the contingency of the readings. Consequently, it is stated that knowing Φ, the readings S_i are independent. Φ is the cause of the readings and, knowing the cause, the consequences are independent. Indeed, this is a very strong hypothesis, far from being always satisfied. However it gives very often satisfactory results and has the main advantage of reducing considerably the complexity of the computation.
- The distributions $\mathbf{P}(S_i \mid \Phi)$ are called "sensor models". Indeed, these distributions encode the way a given sensor responds to the observed phenomenon. When dealing with industrial sensors this is the kind of information directly provided by the device manufacturer. Hovever, these distributions may also very easily be identified by experimenting.
- The most common question asked to this fusion model is $\mathbf{P}(\Phi \mid S_1 \wedge ..... \wedge S_N)$. It should be noticed that it is an inverse question. The capacity to easily answer such inverse questions is one of the main advantage of probabilistic modelling. This is not the only possible question, any question may be asked to this kind of model, and due to the decomposition most of them lead to tractable computations. For instance:
  - $\mathbf{P}(\Phi \mid S_i \wedge S_j) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \mathbf{P}(S_i \mid \Phi) \times \mathbf{P}(S_j \mid \Phi)$ makes the fusion of 2 single sensors. It simplifies nicely as the product of the 2 correponding sensor models.
  - $\mathbf{P}(S_k \mid \Phi \wedge S_i \wedge S_j) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \mathbf{P}(S_i \mid \Phi) \times \mathbf{P}(S_j \mid \Phi) \times \mathbf{P}(S_k \mid \Phi)$ tries to measure the coherence between the readings of 3 sensors and may be used to diagnose the failure of the k^{th} sensor.

3.2.2. Classification

The classification problem may be seen as the same one than the sensor fusion just described. Usually, the problem is called a classification problem when the possible value for Φ is limited to a small number of classes and is called a sensor fusion problem when Φ can be interpreted as a "measure".

A slightly more subtle definition of classification uses one more variable. In this model there are both the variable Φ used to merge the information and C to classify the situation. C has much less values than Φ and it is possible to specify $\mathbf{P}(\Phi \mid C)$ which explicits for each class the possible values of Φ. Answering the classification question $\mathbf{P}(C \mid S_1 \wedge ..... \wedge S_N)$ supposes to sum over the different values of Φ.

The Bayesian program then obtained is as follows:

$$\text{Prog} \begin{cases} \text{Desc} \begin{cases} \text{Spec} \begin{cases} \text{Pertinent Variables} \\ C, \Phi, S_1, ..., S_N \\ \text{Decomposition} \\ \mathbf{P}(C \wedge \Phi \wedge S_1 \wedge ..... \wedge S_N) = \mathbf{P}(C) \times \mathbf{P}(\Phi \mid C) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi) \\ \text{Forms} \\ \textit{Any} \end{cases} \\ \text{Ident} \end{cases} \\ \text{Ques} \\ \mathbf{P}(C \mid S_1 \wedge ..... \wedge S_N) = \frac{1}{Z} \times \sum_{\Phi} \left[\mathbf{P}(C) \times \mathbf{P}(\Phi \mid C) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi) \right] \end{cases} \quad [26]$$

3.2.3. Pattern Recognition

Pattern Recognition is still the same problem than the 2 preceeding ones. However, it is called *recognition* because the emphasis is put on deciding a given value for C rather than getting the distribution $\mathbf{P}(C \mid S_1 \wedge ..... \wedge S_N)$.

Consequently, the pattern recognition community usually does not make a clear separation between the probabilistic inference part of the reasonning and the decision part using a utility function (see section 4.5). Both are considered as a single and integrated decision process.

A reference work on pattern recognition is still Duda's book entitled *Pattern Classification and Scene Analysis* (Duda & Hart, 1973).

3.2.4. Sequences Recognition

The problem is to recognize a sequence of states knowing a sequence of observations and, possibly, a final state.

In section 3.1.2 we presented the Hidden Markov Models (HMM) as a special case of Bayesian filters. These HMMs have been specially designed for sequence recognition, that is why the most common question asked to these models is $\mathbf{P}(S^1 \wedge S^2 \wedge ..... \wedge S^{t-1} \mid S^t \wedge O^0 \wedge ..... \wedge O^t)$ (see equation [17]). That is also why a specialized inference algorithm has been conceived to answer this specific question (see section 4.3.4).

3.2.5. Markov Localization

Another possible variation of the HMM formalism is to add a control variable A to the system. This extension is sometimes called input-output HMM (Bengio & Frasconi, 1995, Cacciatore & Nowlan, 1994, Ghahramani, 2001, Meila & Jordan, 1996), but, in the field of robotics, it has received more attention under the name of Markov Localization (Burgard et al., 1996, Thrun, Burgard & Fox, 1998). In this field, such an extension is natural, as the robot can observe its state by sensors, but can also influence its state via motor commands.

Starting from a HMM structure, the control variable is used to refine the transition model $\mathbf{P}(S^i \mid S^{i-1})$ of the HMM into $\mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})$, which is then called the *action model*. The rest of the HMM is unchanged. The Bayesian progam then obtained is as follows:

$$\text{Prog}\begin{cases}\text{Desc}\begin{cases}\text{Spec}\begin{cases}\text{Pertinent Variables}\\ S^0, \ldots\ldots, S^t, A^0, \ldots\ldots, A^{t-1}, O^0, \ldots\ldots, O^t\\ \text{Decomposition}\\ \mathbf{P}(S^0 \wedge \ldots\ldots \wedge S^t \wedge A^0 \wedge \ldots\ldots \wedge A^{t-1})\\ = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{t} [\mathbf{P}(A^{i-1}) \times \mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\ \text{Forms}\\ \textit{Tables}\end{cases}\\ \text{Ident}\end{cases}\\ \text{Ques}\\ \mathbf{P}(S^t \mid A^0 \wedge \ldots\ldots \wedge A^{t-1} \wedge O^0 \wedge \ldots\ldots \wedge O^t)\\ = \frac{1}{Z} \times \mathbf{P}(A^{t-1}) \times \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1} \wedge A^{t-1}) \times \mathbf{P}(S^{t-1} \mid A^0 \wedge \ldots\ldots \wedge A^{t-2} \wedge O^0 \wedge \ldots\ldots \wedge O^{t-1})]\end{cases} \quad [27]$$

The resulting model is used to answer the question $\mathbf{P}(S^t \mid A^0 \wedge \ldots\ldots \wedge A^{t-1} \wedge O^0 \wedge \ldots\ldots \wedge O^t)$, which estimates the state of the robot, given past actions and observations: when this state represents the position of the robot in its environment, this amounts to localization.

A reference paper to Markov Localization and its use in robotics is Thrun's survey entitled Probabilistic Algorithms in Robotics (Thrun, 2000).

3.2.6. MDP & POMDP

Partially Observable Markov Decision Process

POMDPs are used to model a robot that has to plan and to execute a sequence of actions. A complete review of POMDPs and MDPs by Boutilier et al. (Boutilier, Dean & Hanks, 1999) is an interesting starting point.

Formally, POMDPs use the same probabilistic model than Markov Localization except that it is enriched by the definition of a *reward (and/or cost) function*.

This reward function $\mathbf{R}$ models which states are good for the robot, and which actions are costly. In the most general notation, it therefore is a function that associates, for each couple state - action, a real valued number: $\mathbf{R}, S^i \otimes A^i \rightarrow \Re$.

The reward function also helps driving the planning process. Indeed, the aim of this process is to find an optimal plan in the sense that it maximizes a certain measure based on the reward function. This measure is most frequently the expected discounted cumulative reward:

$$\langle \sum_{t=0}^{\infty} \gamma^t \times R^t \rangle \quad [28]$$

where γ is a discount factor (less than 1), R^t is the reward obtained at time t, and $\langle\ \rangle$ is the mathematical expectation. Given this measure, the goal of the planning process is to find a optimal mapping from probability distributions over states to actions (a *policy*). This planning process, which leads to intractable computation, is sometimes approximated using iterative algorithms called policy iteration or value iteration. These algorithms start with random policies, and improve them at each step until some numerical convergence criterion is met. Unfortunately, state-of-the-art implementations of these algorithms still cannot cope with state spaces of more than a hundred states (Pineau & Thrun, 2002).

An introduction to POMDPs is proposed by Kaelbling et al. (Kaelbling, Littman & Cassandra, 1998).

Markov Decision Process

Another class of approach for tackling the intractability of the planning problem in POMDPs is to suppose that the robot knows what state it is in. The state becomes observable, therefore the observation variable and model are not needed anymore: the resulting formalism is called a (Fully Observable) Markov Decision Process (MDP), and is summed up by the following Bayesian program:

$$\text{Prog}\begin{cases}\text{Desc}\begin{cases}\text{Spec}\begin{cases}\text{Pertinent Variables}\\ S^0, ..., S^t, A^0, ..., A^{t-1}\\ \text{Decomposition}\\ \mathbf{P}(S^0 \wedge ..... \wedge S^t \wedge A^0 \wedge ..... \wedge A^{t-1}) = \mathbf{P}(S^0) \times \prod_{i=1}^{t} [\mathbf{P}(A^{i-1}) \times \mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})]\\ \text{Forms}\\ \textit{Tables}\end{cases}\\ \text{Ident}\end{cases}\\ \text{Ques}\\ \mathbf{P}(A^0 \wedge ..... \wedge A^{t-1} \mid S^t \wedge S^0)\end{cases} \quad [29]$$

- The variables are: $\{S^0, ..., S^t\}$ a temporal sequence of states and $\{A^0, ..., A^{t-1}\}$ a temporal sequence of actions.
- The decomposition makes a first order Markov assumption by specifying that state at time t depends on state at time t-1 and also on action taken at time t-1.
- $\mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})$ is usually represented by a matrix and is called the "transition matrix" of the model.

MDPs can cope with planning in state-spaces bigger than POMDPs, but are still limited to some hundreds of states. Therefore, many recent research efforts are aimed toward hierarchical decomposition of the planning and modelling problems in MDPs, especially in the robotic field, where their full observability hypothesis makes their practical use difficult (Hauskrecht et al., 1998; Lane & Kaelbling, 2001; Pineau & Thrun, 2002 & Diard, 2003).

3.2.7. Bayesian Robot Programming

In his Ph.D. thesis, Olivier Lebeltel (see Lebeltel, 1999 and Lebeltel et al., 2003) proposes a methodology for programming robots taking into account incompleteness and uncertainty. This methodology is called BRP, for Bayesian Robot Programming. The capacities of this programming method have been demonstrated through a succession of increasingly complex experiments. Starting from the learning of simple reactive behaviours, this work proposes probabilistic models of behaviour combination, sensor fusion, hierarchical behaviour composition, situation recognition and temporal sequencing. This series of experiments comprises the steps in the incremental development of a complex robotic program.

As an illustration of this approach, let us present a single example of these various problem dependent probabilistic models, called behaviour combination.

Combination

Suppose that we have defined or learned N different reactive behaviours, each one coded by a Bayesian program $\mathbf{P}(A \wedge O_i \mid \pi_i)$. It is possible to build a new Bayesian program as the *combination* of these N simpler ones. It is defined by:

$$
\text{Prog}\begin{cases}
\text{Desc}\begin{cases}
\text{Spec}\begin{cases}
\text{Pertinent Variables}\\
O_1, \ldots\ldots, O_N, A, H\\
\text{Decomposition}\\
\mathbf{P}(O_1 \wedge \ldots\ldots \wedge O_N \wedge A \wedge H)\\
= \mathbf{P}(O_1 \wedge \ldots\ldots \wedge O_N) \times \mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N) \times \mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N \wedge H)\\
\text{Forms}\\
\mathbf{P}(O_1 \wedge \ldots\ldots \wedge O_N)\\
\mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N)\\
\mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N \wedge [H = 1]) \equiv \mathbf{P}(A \mid O_1 \wedge \pi_1)\\
\ldots\ldots\\
\mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N \wedge [H = N]) \equiv \mathbf{P}(A \mid O_N \wedge \pi_N)
\end{cases}\\
\text{Ident}
\end{cases}\\
\text{Ques}\\
\mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N) = \frac{1}{Z} \times \sum_H [\mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N) \times \mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N \wedge H)]
\end{cases}
\qquad [30]
$$

- $\{O_1, \ldots\ldots, O_N\}$ is a set of N sensory variables, A is a motor variable and H is a selection variable.
- $\mathbf{P}(O_1 \wedge \ldots\ldots \wedge O_N)$ is a prior on the sensory variables, usually uniform.
- $\mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N)$ is a probabilistic model of which behaviour among the N possibles is the most appropriate in a given sensory situation.
- $\mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N \wedge H)$ is a selector which for each possible value of H selects one of the N behaviours.
- The question asked to this behaviour combination is $\mathbf{P}(A \mid O_1 \wedge \ldots\ldots \wedge O_N)$: what is the distribution on the motor variable knowing all the sensory inputs. As H is unknown in this question, the answer is computed by summing on the different possible values of H.

A first point of view on this combination is to consider that we have a probabilistic "if-then-else" or "case". Would the distribution $\mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N)$ be a collection of Diracs, we would have a regular "case" selecting one and only one value of H (one behaviour) for each sensory situation. As $\mathbf{P}(H \mid O_1 \wedge \ldots\ldots \wedge O_N)$ may be more subtle by expressing relative probabilities for the different values of H, we obtain instead a weighted sum of the different behaviours.

Another point of view on this behaviour combination is to consider that we define a special case of mixture models (see section 3.1.3) where the mixed models are the N original reactive behaviours.

3.2.8. Bayesian CAD Modelling

Taking uncertainties into account in CAD systems is a major challenge for this software industry. Since the work of Taylor (Taylor, 1976), in which geometric uncertainties were taken into account in the manipulator planning process for the first time, numerous approaches have been proposed to model these uncertainties explicitly.

Gaussian models to represent geometric uncertainties and to approximate their propagation have been proposed in manipulator programming (Puget, 1989) as well as in assembly (Sanderson, 1997). Gaussian model-based methods have the advantage of requiring few computations. However, they are only applicable when a linearization of the model is possible, and are unable to take into account inequality constraints.

Geometric constraint-based approaches (Taylor, 1976; Owen, 1993) using constraints solvers have been used in robotic task-level programming systems. Most of these methods do not represent uncertainties explicitly. They handle uncertainties using a least-square criterion when the solved constraints systems are over-determined. In the cases where uncertainties are explicitly taken into account (as is the case in Taylor's system), they are described solely as inequality constraints on possible variations.

In his Ph.D. thesis, Kamel Mekhnacha (see Mekhnacha, 1999; Mekhnacha, Mazer & Bessière, 2000; Mekhnacha, Mazer & Bessière, 2001) proposes a Bayesian generic model of CAD modelling.

The main idea of Bayesian CAD modelling is to generalize the constraint-based approaches by taking into account the uncertainties on models. A constraint on a relative pose between two objects is represented by a probability distribution on parameters of this pose instead of a simple equality or inequality. After the specification phase, in which a *probabilistic kinematic graph*, the joint distribution on the

parameters of the model is constructed. For a given problem, the marginal distribution on the unknown parameters is inferred using probability calculus. The original problem is reduced to an optimization problem over the marginal distribution to find a solution with a maximum probability.

A complete Bayesian Program formulation of this approach may be expressed but is far too long to be presented here (see Mekhnacha, 1999).

3.2.9. Biologically Inspired Probabilistic Models

This is of course of main interest for the BIBA project. Indeed, it is so important that 2 specific state of the art cover these aspects:

- Deliverable 6: The biological and neural handling of probabilities
- Deliverable 7: Probabilistic models of perception and action

4. BAYESIAN REASONNING

4.1. Stating the problem

Given the joint distribution $\mathbf{P}(X^1 \wedge X^2 \wedge \ldots \wedge X^n) = \mathbf{P}(L^1) \times \mathbf{P}(L^2 \mid R^2) \times \mathbf{P}(L^3 \mid R^3) \times \ldots \times \mathbf{P}(L^k \mid R^k)$, it is always possible to compute any possible question $\mathbf{P}(Searched \mid Known)$, using the following general inference:

$$\begin{aligned}
\mathbf{P}(Searched \mid Known) &= \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \mid Known) \\
&= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known)}{\mathbf{P}(Known)} \\
&= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known)}{\sum_{Searched}\left[\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known)\right]} \\
&= \frac{1}{Z} \times \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known) \\
&= \frac{1}{Z} \times \sum_{Unknown}\left[\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i)\right]
\end{aligned} \qquad [31]$$

where the first equality results from the marginalization rule (equation [7]), the second results from the product rule (equation [5]) and the third corresponds to a second application of the marginalization rule. The denominator appears to be a normalization term. Consequently, by convention, we will replace it by Z. It is finally possible to replace the joint distribution by its decomposition (see equation [10]).

It is well known that general Bayesian inference is a very difficult problem, which may be practically intractable. Exact inference has been proved to be NP-hard (Cooper, 1990) and the general problem of approximate inference too (Dagum & Luby, 1993). Numerous heuristics and restrictions to the generality of the possible inferences have been proposed to achieve admissible computation time. The purpose of this section is to make a short review of these heuristics and techniques.

Two main problems have to be solved: searching the modes in a high dimensional space, and marginalizing in a high dimensional space.

Since *Searched* may be a conjunction of numerous variables, each of them possibly having a lot of values or even being continuous, it is seldom possible to exhaustively compute $\mathbf{P}(Searched \mid Known)$. One may then decide either to build an approximate representation of this distribution or to directly sample from this distribution. In both cases the challenge is to find the modes where most of the probability density is concentrated. This may be very difficult, as most of the probability may be concentrated in very small sub spaces of the whole searched space. Searching the modes of a distribution in a high dimensional space will be the subject of section 4.3.

The situation is even worse, as computing the value of $\mathbf{P}(Searched \mid Known)$ for a given value of *Searched* (a single point of the searched space of the preceeding paragraph) is by itself a difficult problem. Indeed, it supposes to marginalize the joint distribution on the space defined by *Unknown*. *Unknown* (like *Searched*) may be a conjunction of numerous variables, each of them possibly having a lot of values or even being continuous. Consequently, the sum should also be either approximated or sampled. The challenge is then to find the modes of

$$\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \qquad [32]$$

(on the search space defined by *Unknown*), where most of the probability density is concentrated and which mostly contribute to the sum. Finally, marginalizing in a high dimensional space appears to be a very similar problem to searching the modes in a high dimensional space. This will appear even more clearly in section 4.4, which will treat of marginalization.

However, before treating these numerical problems it is usually possible to make some symbolic simplifications of

$$\frac{1}{Z} \times \sum_{Unknown} \left[\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \right] \qquad [33]$$

This simplification phase may drastically reduce the number of sums necessary to compute the distribution $\mathbf{P}(Searched \mid Known)$. The next section (4.2) presents the different simplification possibilities.

4.2. Simplification

4.2.1. Question Independent Simplifications

The basic inference done in Bayesian Networks consists in computing $\mathbf{P}(X_i \mid Known)$. The values of the different variables appearing in *Known* are called "evidences". Of course the set and values of *Known* variables may vary.

A Bayesian Network is defined by the decomposition:

$$\mathbf{P}(X_1 \wedge ..... \wedge X_N) = \prod_{i=1}^{N} \mathbf{P}(X_i \mid R_i). \qquad [34]$$

This decomposition encodes the logical dependencies between the X_i variables. However, it may not be an interesting decomposition from a computing point of view because it may lead to very intensive calculations for the questions $\mathbf{P}(X_i \mid Known)$.

Simplifying the computation in Bayesian Networks supposes to find another decomposition $\prod_{j=1}^{M} \mathbf{P}(\tilde{L}_j \mid \tilde{R}_j)$ which reduces the computational burden but stays equivalent to the defining one:

$$\begin{aligned} \mathbf{P}(X_1 \wedge ..... \wedge X_N) &= \prod_{i=1}^{N} \mathbf{P}(X_i \mid R_i) \\ &= \prod_{j=1}^{M} \mathbf{P}(\tilde{L}_j \mid \tilde{R}_j) \\ &= \prod_{j=1}^{M} \frac{\mathbf{P}(\tilde{L}_j \wedge \tilde{R}_j)}{\mathbf{P}(\tilde{R}_j)} \\ &= \prod_{j=1}^{M} \frac{\mathbf{P}(\tilde{L}_j \wedge \tilde{R}_j)}{\sum_{R_j} \mathbf{P}(\tilde{L}_j \wedge \tilde{R}_j)} \end{aligned} \qquad [35]$$

The last term of this equation makes clear that we want both to keep the dimension of $\tilde{L}_j \wedge \tilde{R}_j$ as small as possible (to reduce the needed memory), and also to keep the dimension of $\tilde{R}_j$ as small as possible (to reduce the amount of necessary summations).

The junction tree algorithm, also often called JLO after its inventors (see Jensen, Lauritzen & Olesen, 1990), searches for such a decomposition and implements the corresponding computation to solve $\mathbf{P}(X_i \mid Known)$.It is a generalization of the orginal Pearl's message-passing algorithm, used to update probabilities in polytree shaped Bayesian networks (see Pearl, 1982).

The main idea of the junction tree algorithm is to convert the Bayesian network into a tree by clustering the nodes together. After building this tree of clusters, inference can be done efficiently by a single message passing algorithm.

The junction tree algorithm is decomposed in two phases: the construction of the junction tree which has to be run only once for a given Bayesian network, and an evidence propagation algorithm which has to be run once for each inference.

The construction of the junction tree needs 3 steps:

1 Construction of the "moral" graph, by linking together each pair of parents for every node which has at least 2 parents and transformation of the directed graph into an undirected one.
2 Triangulation of the graph.
3 Transformation of the graph into a junction tree by using the Kruskal algorithm to compute a maximal weight spanning tree using the cliques extracted form the triangulated graph. It should be noted that the Kruskal algorithm is only a heuristic and is not guaranteed to find the optimum junction tree.

The evidence propagation algorithm needs 4 steps:

1 Initializing the probability distributions of the junction tree.
2 Entering evidences into the junction tree.
3 Propagating evidences throughout the junction tree.

4 Possibly updating the Bayesian network probability distribution.

All the details of this popular algorithm may be found in the already cited text books (see section 3.1.1) and also in Jensen's book (Jensen, 1996).

It is very important to note that this JLO algorithm and the corresponding decomposition is very efficient for computing questions of the form $\mathbf{P}(X_i \mid Known)$, but could be very inefficient for computing the solutions to other questions: for example, questions that involve several variables in the *Searched* conjunction: $\mathbf{P}(X_i \wedge ..... \wedge X_k \mid Known)$. For such questions, supplementary summations are required and the cliques structure found by JLO may be terrible for these summations.

4.2.2. Question Dependent Simplifications

The junction tree simplification just presented tries to find a decomposition of the joint distribution (i.e. a junction tree), which reduces the global computational burden.

However, taking into account the question at hand may lead to much better simplifications. The purpose of this section is to describe these question dependent simplifications.

These simplifications can be applied one after another (in the order they are described) for a maximum efficiency. Furthermore, they are not restricted to Bayesian networks and are valid for any kind of Bayesian decomposition.

Simple Simplification

Let us start back from the basic equation:

$$\mathbf{P}(Searched \mid Known) = \frac{1}{Z} \times \sum_{Unknown} \left[\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \right] \qquad [36]$$

Considering the different terms of the product $\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i)$, three possibilities of obvious simplifications may appear:

1 When a term is a uniform distribution it can be simplified: it vanishes from the expression and its value will implicitly be taken into account in the normalization constant Z.

2 When a term is a distribution where all the variables have *Known* values, then it is a constant for this question and may also be simplified.

3 When a term is a distribution where all the variables are either *Searched* or *Known*, then it can be factorized out of the sum.

Finally, we get a new expression of the form:

$$\mathbf{P}(Searched \mid Known) = \frac{1}{Z} \times \prod_{j \in J} \mathbf{P}(L^j \mid R^j) \times \sum_{Unknown} \left[\prod_{i \in I} \mathbf{P}(L^i \mid R^i) \right] \qquad [37]$$

Elimination of Distributions That Sum to One

Now, considering $\sum_{Unknown} \left[\prod_{i \in I} \mathbf{P}(L^i \mid R^i) \right]$, we can try to find an order on the sum to simplify terms that sum to 1.

Indeed, when a term $\mathbf{P}(L^i \mid R^i)$ appears in the sum, if all the variables appearing in L^i are part of *Unknown* (summed) and if all the variables appearing in R^i are either part of *Known* or *Unknown*, then $\mathbf{P}(L^i \mid R^i)$ sums to 1 and vanishes out of the global sum.

This operation often leads to impressive simplifications. In equation [16] this is how we simplified line 4 into line 5:

$$\sum_{S^0, ..., S^{t-1}} \left[\frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge ..... \wedge O^t)} \times \sum_{\substack{O^{t+1}, ..., O^N \\ S^{t+1}, ..., S^N}} \left[\prod_{j=t+1}^{N} [\mathbf{P}(S^j \mid S^{j-1}) \times \mathbf{P}(O^j \mid S^j)] \right] \right] \qquad [38]$$

$$= \sum_{S^0, ..., S^{t-1}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge ..... \wedge O^t)}$$

General Distributive Law

Finally, the last simplification that can be made is to reorder the sums of $\sum_{Unknown}\left[\prod_{i\in I}\mathbf{P}(L^i \mid R^i)\right]$, in order to minimize the number of operations to make.

The algorithm corresponding to this simplification is very similar to the junction tree algorithm described just above. Whereas the JLO algorithm does not take into account the question, the general distributive algorithm takes explicit care of this question.

Roughly, *Searched* is considered as a new node in the network and the moralisation, triangulation and construction of the cliques of the junction tree are made taking into account this new node. $\mathbf{P}(Searched \mid Known)$ may then be considered and treated as a $\mathbf{P}(X_i \mid Known)$ question.

All the details on the general distributive law may be found in a paper by Aji and McEliece (Aji & McEliece, 2000).

4.3. Searching the Modes in High-Dimensional Spaces

4.3.1. Building Explicit Representations of the Distribution

Representation using a sample set of points

Suppose we are interested in building an explicit representation of a given D-dimensional distribution $\mathbf{P}(X)$.

The most simple representation one can imagine is to encode $\mathbf{P}(X)$ as a sample set of N points:

$$\mathbf{P}(X) \equiv \{x_1, ..., x_N\} \quad [39]$$

This kind of representation, even simple, can be very useful if this distribution is used in a sampling process *directly* or *indirectly.*

We say that $\mathbf{P}(X)$ is used *directly* in a sampling process if the problem consists in drawing a set of M points $\{x_1, ..., x_M\}$.

We say that $\mathbf{P}(X)$ is used *indirectly* in a sampling process if the problem consists in estimating a distribution $\mathbf{P}(Y)$ as an integral on the X space using a Monte Carlo method:

$$\mathbf{P}(Y) = \sum_X \mathbf{P}(X) \times \mathbf{P}(Y \mid X) \quad [40]$$

Drawing a point directly from $\mathbf{P}(X)$ consists in drawing uniformly an integer value i between 1 and N and returning the corresponding point in the set $\{x_1, ..., x_N\}$. This process may be iterated M times to get M points.

If $\mathbf{P}(X)$ is used indirectly then $\mathbf{P}(Y)$ may be estimated using a Monte Carlo integration method (see section 4.4.2):

$$\mathbf{P}(Y) \approx \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{P}(Y \mid [X = x_i]) \quad [41]$$

This kind of representation can be easily improved by encoding $\mathbf{P}(X)$ as a sample set of N couples:

$$\mathbf{P}(X) \equiv \{(x_1, \omega_1), ..., (x_N, \omega_N)\} \quad [42]$$

where the ω_i are weight values, proportional to $\mathbf{P}(x_i)$.

This kind of representation is especially used for Particle Filters (see section 3.1.2) where $\mathbf{P}(X)$ is used for estimating a sum, using an importance sampling Monte Carlo Method (see section 4.4.2):

$$\mathbf{P}(Y) \approx \frac{\sum_{i=1}^{N} [\omega_i \times \mathbf{P}(Y \mid [X = x_i])]}{\sum_{i=1}^{N} \omega_i} \quad [43]$$

Both kind of representation clearly suffers for important limitations:

- It is obviously very difficult to represent huge high-dimensional spaces with a reasonable number of points.
- It is very difficult to place the points at the modes where they are significant.

- It is clear that these methods of representation are unable to make generalization in the neighborhood of the points of the sample set: $\mathbf{P}(X)$ can only be evaluated for these points.

Multi Resolution Binary Tree (MRBT)

Unlike the preceeding representations, a Multi-Resolution Binary Tree (MRBT) is an explicit representation having a capacity of generalization. Using a MRBT, we can compute, for all point x, the probability value $\mathbf{P}([X = x])$.

The main idea of MRBTs is that high-probability regions of the distribution $\mathbf{P}(X)$ should be represented with a high resolution and low-probability regions may be represented with a low resolution.

A MRBT is built incrementally by inserting N pairs $\{(x_1, \mathbf{P}([X = x_1])), ..., (x_N, \mathbf{P}([X = x_N]))\}$. This set of pairs is generated using an external process such as, for instance, a Genetic algorithm or a Metropolis sampler.

When inserting a given pair $(x_i, \mathbf{P}([X = x_i]))$, the binary tree is updated so that the resolution of the region including the point x_i is increased by splitting the corresponding node on one dimension.

The built MRBT can be used to get the probability value of any given point x (i.e. to compute $\mathbf{P}([X = x])$) by searching (using dichotomy) the leaf node corresponding to x and returning its probability value.

To draw a point from the distribution $\mathbf{P}(X)$, we also use dichotomy to draw a leaf node. Starting from the root node, we choose to go to its first child or to the second one according to their respective total probability values. This procedure is iterated until a leaf node is reached. Having this leaf node, drawing a value x from $\mathbf{P}(X)$ consists in a uniform draw in the region encoded by this leaf.

More details on MRBT implementation and use can be found in a patent protecting this method (see Bessière, 2002).

An obvious perspective to enhance the MRBT approach would be to use wavelet coding for the probability distributions. We are investigating the litterature too see if work has already been done in this direction and plan to follow this research track during the execution of the BIBA project.

4.3.2. Distribution Sampling Methods

The problem of drawing samples from a given distribution $\mathbf{P}(X)$ is still a challenging one, especially in high D-dimensional spaces.

Direct Sampling methods

Direct Sampling from Normal (Gaussian) Distributions

In some cases, it is possible, when disposing of a uniform random generator, to use a *transformation function* to sample a given distribution $\mathbf{P}(X)$ (see Rubinstein, 1981).

One of the more important transformation functions is the Box-Muller transformation (Box & Muller, 1958). This transformation permits to generate a set of random numbers from a one-dimensional Normal distribution using another set of random numbers generated by a uniform random generator.

Sampling a multi-dimensional Normal distribution $\mathbf{P}(X) = \mathbf{G}(X, \mu, \sigma)$ is also possible by diagonalizing the variance matrix Σ. By diagonalization, the obtained major axes (eigenvectors) define a new coordinate system (in which the components are independent). Therefore, the problem is reduced to drawing each component independently from D one-dimensional Normal distributions having, for each component, the eigenvalue v_i as variance. The obtained vector $x^{Diag} = [x_1^{Diag}, ..., x_D^{Diag}]$ is then rewritten in the original coordinate system by multiplying it by the transpose of the eigenmatrix (having eigenvectors as columns) to get the sample vector $x = [x_1, ..., x_D]$.

Direct Sampling Using Repartition Function

When disposing (analytically or numerically) of the repartition function $\mathbf{F}(X)$ corresponding to a target distribution $\mathbf{f}(X)$:

$$\mathbf{F}(X) = \int_{-\infty}^{X} \mathbf{f}(i)di \qquad [44]$$

the problem of sampling $\mathbf{f}(X)$ is reduced to the problem of inverting the repartition function $\mathbf{F}(X)$. Drawing a sample point from $\mathbf{f}(X)$ consists in:

1 drawing an uniform random value $u \in [0, 1]$;

2 getting the drawn point as $x = \mathbf{F}^{-1}(u)$.

In the general case, analytical forms of the repartition function $\mathbf{F}(X)$ and its inverse are not available. Explicitly computing the repartition function $\mathbf{F}(X)$ and inversing it numerically is possible for low-dimensional spaces. It is usually impossible in practice for high-dimensional cases.

Monte Carlo Methods

Monte Carlo Methods group together several different methods for sampling in high dimensional spaces. In this section we present some of the most popular variants: importance sampling, rejection sampling, Metropolis sampling and Gibbs sampling.

Two excellent starting points on Monte Carlo Methods are the tutorials by Neal (Neal, 1993) and MacKay (MacKay, 1996).

Importance Sampling

Suppose we are interested in sampling a distribution $\mathbf{P}(X)$ for which no direct sampling method is available and that we are able to evaluate this distribution for each point x_i of the state space.

Suppose also that we have a simpler distribution $\mathbf{Q}(X)$ (called *proposal distribution*) that we can also evaluate for each point x_i and for which a direct sampling method is available.

Using *Importance Sampling* to sample $\mathbf{P}(X)$ consists in generating N pairs $\{(x_1, \omega_1), ..., (x_N, \omega_N)\}$ where the points x_i are drawn from $\mathbf{Q}(X)$ and where:

$$\omega_i = \frac{\mathbf{P}(x_i)}{\mathbf{Q}(x_i)} \qquad [45]$$

This sampling method is especially used for Monte Carlo Importance Sampling integration method (see section 4.4.2).

Rejection Sampling

Suppose we are interested in sampling a distribution $\mathbf{P}(X)$ for which no direct sampling method is available and that we are able to evaluate this distribution for each point x_i of the state space.

Suppose also that we have a simpler distribution $\mathbf{Q}(X)$ that we can evaluate for each point x_i, for which a direct sampling method is available, and that respects the constraint:

$$\exists c, \forall x, [c \times \mathbf{Q}(x) > \mathbf{P}(x)] \qquad [46]$$

Using *Rejection Sampling* to draw a point of $\mathbf{P}(X)$ consists in drawing a point x_i from $\mathbf{Q}(X)$ and accepting it with a probability of:

$$\frac{c \times \mathbf{Q}(x_i)}{\mathbf{P}(x_i)} \qquad [47]$$

The complete procedure is then:

1 Draw a candidate point x_i from $\mathbf{Q}(X)$;

2 Evaluate $c \times \mathbf{Q}(x_i)$;

3 Generate an uniform random value $u \in [0, c \times \mathbf{Q}(x_i)]$;

4 if $\mathbf{P}(x_i) > u$ then the point x_i is accepted, otherwise, the point is rejected.

It is clear that this rejection sampling will be efficient if the distribution $\mathbf{Q}(X)$ is a good approximation of $\mathbf{P}(X)$. Otherwise, the rejection rate will be very important.

Metropolis Sampling

Previous methods using a *proposal distribution* perform well if $\mathbf{Q}(X)$ is a good approximation of $\mathbf{P}(X)$. For complex problems in high-dimensional spaces, it is difficult to find such a distribution.

The Metropolis algorithm (Metropolis et al., 1953) uses a Markovian process in which a sequence of states x^t are generated. The new state x^t depends of the previous one x^{t-1}. This algorithm is an example of *Markov Chains Monte Carlo* (MCMC) methods.

Instead of using a single proposal distribution $\mathbf{Q}(X)$, the Metropolis algorithm uses a proposal distribution $\mathbf{Q}(X_i, X^t)$, which depends of the current state x^t. This distribution can be a simple distribution (a Normal distribution having x^t as mean value for example).

Suppose the current state is x^t. A candidate x_i is generated from $\mathbf{Q}(X_i, X^t)$. To accept or reject this candidate we have to compute:

$$a = \frac{\mathbf{P}(x_i) \times \mathbf{Q}(x_i, x^t)}{\mathbf{P}(x^t) \times \mathbf{Q}(x^t, x_i)} \quad [48]$$

If $a > 1$ then x_i is accepted. Otherwise, it is accepted with probability a. If x_i is accepted, we set $x^{t+1} = x_i$. If x_i is rejected, then we set $x^{t+1} = x^t$.

A frequent choice for the proposal distribution $\mathbf{Q}(X_i, X^t)$ is to chose a symmetrical one (a Normal distribution having x^t as mean value for example), then:

$$\frac{\mathbf{Q}(x_i, x^t)}{\mathbf{Q}(x^t, x_i)} = 1 \quad [49]$$

and we get:

$$a = \frac{\mathbf{P}(x_i)}{\mathbf{P}(x^t)} \quad [50]$$

One drawback of MCMC techniques is that we must in general wait for the chain to reach equilibrium. This can take a long time, and it is sometimes difficult to tell when it happens.

Gibbs Sampling

Gibbs sampling, also known as *heatbath method*, is another example of a Markov Chain Monte Carlo sampling technique. It has come into prominence only recently with the works of Geman and Smith (Geman & Geman, 1984 and Smith & Roberts, 1993). It is a method for sampling from distributions over at least two dimensions. It can be viewed as a Metropolis method in which the proposal distribution $\mathbf{Q}(X)$ is defined in terms of the conditional distributions of the joint distribution $\mathbf{P}(X)$. It is assumed that while $\mathbf{P}(X)$ is too complex to draw samples from directly, its conditional distributions $\mathbf{P}(X_i \mid X_1 \wedge ..... \wedge X_{i-1} \wedge X_{i+1} \wedge ..... \wedge X_N)$ are tractable.

In a general case of a system of N variables, a single iteration involves sampling one variable at a time:

$$\begin{aligned} x_1^{t+1} &\sim \mathbf{P}(X_1 \mid x_2^t \wedge ..... \wedge x_N^t) \\ x_2^{t+1} &\sim \mathbf{P}(X_2 \mid x_1^t \wedge x_3^t \wedge ..... \wedge x_N^t) \\ &..... \\ x_N^{t+1} &\sim \mathbf{P}(X_N \mid x_1^t \wedge ..... \wedge x_{N-1}^t) \end{aligned} \quad [51]$$

4.3.3. Variational Methods

The key to *variational methods* is to convert a probabilistic inference problem into an optimization problem. In this way standard tools of constraint optimization can be use to solve the inference problem. The idea is to replace a joint distribution $\mathbf{P}(X) = \mathbf{P}(X_1 \wedge ..... \wedge X_N)$ (represented by an acyclic graph in the case of a Bayesian Net) by an approximation $\mathbf{Q}(X)$ and to compute the kullback-leibler divergence between the two distributions.

To do that, we consider the energy of a configuration X defined by:

$$\mathbf{E}(X) = -\mathbf{log}(\mathbf{P}(X)) - \mathbf{log}(Z) \quad [52]$$

and the variational free energy (or kullback leibler distance) as:

$$\mathbf{F}(\mathbf{Q}, \mathbf{P}) = \sum_X \left[\mathbf{Q}(X) \times \mathbf{log}\left(\frac{\mathbf{Q}(X)}{\mathbf{P}(X)}\right) \right] - \mathbf{log}(Z) \quad [53]$$

this distance is minimized when $\mathbf{P}(X) = \mathbf{Q}(X)$.

Of course minimizing $\mathbf{F}$ is as difficult as the original inference problem. But by considering different family of $\mathbf{Q}(X)$ we obtained different approximation of $\mathbf{F}$ and as a consequence different variational methods.

For example, if one restricts itself to the family of factorized independant distributions:

$$\mathbf{Q}(X) = \prod_{i=1}^{N} \mathbf{Q}_i(X_i) \quad [54]$$

the variational method boiled down to the *mean field* approximation. Minimizing $\mathbf{F}(\mathbf{Q}, \mathbf{P})$ is greatly

simplified using the acyclic graph stucture of $\mathbf{P}(X)$.

General introduction to variational methods may be found in 2 introductions to the field by Jordan (Jordan et al., 1999; Jordan & Weiss, 2002). Interesting refinements are described in Yedidia's paper (Yedidia, Freeman & Weiss, 2002).

4.3.4. Viterbi Algorithm

The recognition of a sequence in a HMM (defined by [17]) is carried out by the Viterbi algorithm (Forney, 1973), which determines the most likely state sequence given a sequence of observations.

In Hidden Markov Models, many state sequences leading to a state s^T may generate the same observed sequence $\{o^0, ..., o^T\}$. Given one such output sequence, we are interested in determining the most likely state sequence $\{s^0, ..., s^{T-1}\}$ that could have generated the observed sequence. The corresponding question is:

$$\mathbf{P}(S^0 \wedge S^2 \wedge ..... \wedge S^{T-1} \mid s^T \wedge o^0 \wedge ..... \wedge o^T) \qquad [55]$$

The most likely state sequence is found by using the probability of the partial alignment ending at state i at time t:

$$\delta_t(i) = \mathbf{Max}_{S^1 \wedge S^2 \wedge ..... \wedge S^{t-1}}(\mathbf{P}(S^0 \wedge S^2 \wedge ..... \wedge S^{t-1} \mid s^T \wedge o^0 \wedge ..... \wedge o^t)) \qquad [56]$$
$$2 \leq t \leq T, 1 \leq i \leq N$$

where N is the size of the state space.

A recursive computation is given by equation:

$$\delta_t(i) = \mathbf{Max}_{1 \leq j \leq N}(\delta_{t-1}(j) \times \mathbf{P}([S^t = i] \mid [S^{t-1} = j])) \times \mathbf{P}(O^t \mid [S^t = i]) \qquad [57]$$
$$2 \leq t \leq T, 1 \leq i \leq N, 1 \leq j \leq N$$

The Viterbi algorithm is a dynamic programming search that computes the best partial state sequence up to time t for all states. The most likely state sequence $\{s^0, ..., s^{T-1}\}$ is obtained by keeping track of back pointers for each computation of which previous transition leads to the maximal partial path probability. By tracing back from the final state, we get the most likely state sequence.

4.4. Marginalization (Integration) in High-Dimensional Spaces

Integral calculus is the basis of Bayesian inference. Unfortunately analytic methods for integral evaluation seem very limited in real-world applications, where integrands may have complex shapes and integrations spaces may have very high dimensionality.

Domain subdivision-based methods (such as trapezoidal or Simpson methods) are deterministic numerical techniques often used for numerical integration in low-dimensional spaces. However, these techniques are poorly adapted for high-dimensional cases. These techniques will not be discussed further in this report but good sources for such techniques are the numerical recipes (Press et al., 1992) and a book by Davis (Davis & Rabinowitz, 1975).

Monte Carlo methods (MC) are powerful stochastic simulation techniques that may be applied to solve optimization and numerical integration problems in large-dimensional spaces. Since their introduction in the physics literature in the 1950s, Monte Carlo methods have been at the center of the recent Bayesian revolution in applied statistics and related fields.

4.4.1. Analytical Integration

Analytical solutions to integration problems are available in well-catalogued instances in which the integrand have particular shapes. The book by Gradshteyn (Gradshteyn & Ryzhik, 1980) is a useful and standard refererence on these methods.

In probabilistic inference, the most well known and interesting particular case is when the integrand is a product of generalized Normals (Dirac's delta functions and Gaussians) and when the model is linear or can be linearized (variance matrices are small enough). If we have to compute the integral:

$$I(Y) = \int_{-\infty}^{\infty} \mathbf{P}(X) \times \mathbf{P}(Y \mid X) \times dX^D, \qquad [58]$$

where:

- $\mathbf{P}(X) = \mathbf{G}(X, \mu_X, \Sigma_X)$
- $\mathbf{P}(Y \mid X) = \mathbf{G}(Y, A \bullet X, \Sigma_Y)$

then we get the analytical solution:

$$I(Y) = \mathbf{G}(Y, [A \bullet \mu_X], [A \bullet \Sigma_X \bullet A^T + \Sigma_Y]), \quad [59]$$

where A is a constant matrix (or a Jacobian in a linearized model) and A^T is its tranpose.

This analytical resolution of the integral is used, for example, extensively in Kalman Filters (see section 3.1.2) and explains their practical sucess.

4.4.2. Monte Carlo Methods for Numerical Integration

The aim of Monte Carlo methods is to approximate efficiently the D-dimensional integral (where D can be very large):

$$I = \int \mathbf{P}(X) \times \mathbf{Q}(X) \times dX^D \quad [60]$$

We will generally assume that X is a D-dimensional vector with real, discrete or real/discrete components. We will use the symbol $\int$..... as a generalized integration operator for both real integrals (over real components) and sums (over discrete components).

Assuming that we cannot visit every single location x_i in the state (integration) space, the simplest solution for estimating the integral [60] is to uniformly sample the integration space X and then estimate I by $\widehat{I}$:

$$\widehat{I} = \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{P}(x_i) \times \mathbf{Q}(x_i) \quad [61]$$

High-dimensional probability distributions are often concentrated on a small region T of the state (integration) space X, known as its *typical set*. For example, if the space X contains a large number of roughly independent variables, then the volume $\lceil T \rceil$ of the region T is given by:

$$\lceil T \rceil \approx 2^{\mathbf{H}(\mathbf{P}(X))} \quad [62]$$

where $\mathbf{H}(\mathbf{P}(X))$ is the Shannon entropy (see MacKay, 1996):

$$\mathbf{H}(\mathbf{P}(X)) = -\sum_{X} \mathbf{P}(X) \times \mathbf{log}(\mathbf{P}(X)) \quad [63]$$

The number N of points drawn uniformly for the state (integration) space X have to be sufficiently large to cover the region T containing most of the probability mass of $\mathbf{P}(x)$:

$$N \propto \frac{\lceil X \rceil}{\lceil T \rceil} \quad [64]$$

where $\lceil X \rceil$ is the volume of the state space. This fact makes the exploration of the state space using uniform sampling very expensive in the general case.

Instead of exploring the integration space uniformly, Monte Carlo methods try to use the information provided by the distribution $\mathbf{P}(X)$ to explore this space more efficiently. The main idea of these techniques is to approximate the integral I by estimating the expectation of the function $\mathbf{Q}(X)$ under the distribution $\mathbf{P}(X)$:

$$I = \int \mathbf{P}(X) \times \mathbf{Q}(X) \times dX^D = \langle \mathbf{Q}(X) \rangle \quad [65]$$

Clearly, if we are able to generate a set of points (vectors) $\{x_1, ..., x_N\}$ from $\mathbf{P}(X)$, the expectation of $\widehat{I}$ is I. Also, as the number of samples N increases, the variance of the estimator $\widehat{I}$ will decrease as σ^2/N where σ^2 is the variance of $\mathbf{Q}$:

$$\sigma^2 = \int \mathbf{P}(X) \times (\mathbf{Q}(X) - \widehat{Q}) \times dx^D \quad [66]$$

and $\widehat{Q}$ is the expectation of $\mathbf{Q}$.

This result is one of the important properties of Monte Carlo methods: the accuracy of Monte Carlo estimates is independent of the dimensionality of the integration space.

A good survey of Monte Carlo sampling techniques is proposed by Neal (Neal, 1993).

Simple (or Perfect) Monte Carlo integration

Suppose we are able to get a set of samples $\{x_1, ..., x_N\}$ from the distribution $\mathbf{P}(X)$, we can use these samples to get the estimator:

$$\widehat{I} = \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{Q}(x_i) \qquad [67]$$

The *Perfect* Monte Carlo method assumes the capacity to sample the distribution $\mathbf{P}(X)$ efficiently. This is possible when $\mathbf{P}(X)$ is a standard and simple distribution with a direct sampling method, or a product of standard distributions on which a direct sampling is possible using Gibbs algorithm (see section 4.3.2).

For instance, if $\mathbf{P}(X) = \mathbf{P}(X_1 \wedge X_2) = \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1)$ where $\mathbf{P}(X_1)$ is a uniform distribution and $\mathbf{P}(X_2 \mid X_1)$ is a Normal distribution having X_1 as mean and a fixed value as variance, then drawing a point $x^t = (x_1^t, x_2^t)$ consists in:

- drawing x_1^t from $\mathbf{P}(X_1)$ (i.e uniformly on the possible values).
- drawing x_2^t from the normal distribution $\mathbf{P}(X_2 \mid [X_1 = x_1^t])$.

Importance Sampling Monte Carlo integration

Suppose now that we are unable to generate sample points directly from the distribution $\mathbf{P}(X)$ but only from a simpler distribution $\mathbf{S}(X)$ called the *sampling distribution*.

Using Importance Sampling (see section 4.3.2), a set of N points is generated from $\mathbf{S}(X)$. If these sample points were generated from $\mathbf{P}(X)$, we could estimate I using equation [67]. But as these points have been generated from $\mathbf{S}(X)$ and not from $\mathbf{P}(X)$, the values of X for which $\mathbf{S}(X)$ is greater than $\mathbf{P}(X)$ will be over-represented and the values for which $\mathbf{S}(X)$ is less than $\mathbf{P}(X)$ will be under-represented. To take into account this problem, *importance sampling* introduce weights:

$$\omega_i = \frac{\mathbf{P}(x_i)}{\mathbf{S}(x_i)} \qquad [68]$$

These weights are used to add the importance of each point in the estimator:

$$\widehat{I} = \frac{1}{\sum_{i=1}^{N} \omega_i} \times \sum_{i=1}^{N} [\omega_i \times \mathbf{Q}(x_i)] \qquad [69]$$

This method of integration is used especially in Particle Filters (see section 3.1.2).

5. BAYESIAN LEARNING AND EVOLUTION

5.1. Problematic

A Bayesian program has the following structure:

$$\texttt{Prog}\left\{\begin{array}{l}\texttt{Desc}\left\{\begin{array}{l}\texttt{Spec}\ (\pi)\left\{\begin{array}{l}\texttt{Pertinent Variables}\\ X = X_1 \wedge \ldots\ldots \wedge X_N\\ \texttt{Decomposition}\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \mid \delta \wedge \pi) = \prod_{i=1}^{M} \mathbf{P}(L_i \mid R_i \wedge \delta \wedge \pi)\\ \texttt{Forms}\\ \mathbf{P}(L_i \mid R_i \wedge \delta \wedge \pi) = \mathbf{F}^i_{\lambda_i}(L^i)\end{array}\right.\\ \texttt{Ident}\ (\delta)\end{array}\right.\\ \texttt{Ques}\end{array}\right. \qquad [70]$$

In section 3 we showed that most probabilistic models can be specified using this formalism. In section 4 we presented how to automate computation of such programs. At this point, the reader should be convinced of the interest of such probabilistic models for both solving important engineering problems and for building interesting models of cognition.

However, an important question has still to be answered: where do the Bayesian programs come from? From the engineering point of view this question is: is there a methodology to develop Bayesian programs? From the scientific point of view this question could be translated as: is there any natural process that could explain the apparition of probabilistic models in the brains of living beings? The purpose of this 5th section is to review the different aspects of these questions.

The global question (where do the Bayesian programs come from?) can be divided into sub-questions, which can be answered separately and can be made mathematically precise:

1 How to identify (learn) the value of the free parameters?
2 How to compare different probabilistic models (specifications)?
3 How to find interesting decompositions and associated parametric forms?
4 How to find the pertinent variables to model a phenomenon?

5.1.1. How to identify (learn) the value of the free parameters?

Given some specification π (consisting of the pertinent variables, the decomposition of the joint distribution and the parametric forms), and given some data set δ, one wants to identify the values of the free parameters that best take into account the data.

It is always possible to make the free parameters appear as an explicit variable Λ of the model:

$$\texttt{Prog}\left\{\begin{array}{l}\texttt{Desc}\left\{\begin{array}{l}\texttt{Spec}\ (\pi)\left\{\begin{array}{l}\texttt{Pertinent Variables}\\ X \wedge \Lambda\\ X = X_1 \wedge \ldots\ldots \wedge X_N\\ \Lambda = \Lambda_1 \wedge \ldots\ldots \wedge \Lambda_M\\ \texttt{Decomposition}\\ \mathbf{P}(X_1 \wedge \ldots\ldots \wedge X_N \wedge \Lambda_1 \wedge \ldots\ldots \wedge \Lambda_M \mid \delta \wedge \pi)\\ = \mathbf{P}(\Lambda \mid \delta \wedge \pi) \times \prod_{i=2}^{M} \mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge \pi)\\ \texttt{Forms}\\ \mathbf{P}(L^i \mid R^i \wedge \Lambda_i \wedge \pi) = \mathbf{F}^i_{\lambda_i}(L^i)\end{array}\right.\\ \texttt{Ident}\ (\delta)\end{array}\right.\\ \texttt{Ques}\end{array}\right. \qquad [71]$$

- Λ_i represents the free parameters of the i-th form and Λ the set of all these free parameters.
- The goal of the parameters is to sum up what has been learned from the data. Consequently:
  - The parameters depend on the data: $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$

° Knowing the parameters, the distributions does not depend any more on the data:

$$\mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge \delta \wedge \pi) = \mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge \pi) \quad [72]$$

Therefore, there is a Bayesian formulation of the parameter identification problem:

$$\texttt{Prog} \begin{cases} \texttt{Desc} \begin{cases} \texttt{Spec } (\pi) \begin{cases} \texttt{Pertinent Variables} \\ \Delta \wedge \Lambda \\ \Delta = {}_1X \wedge \ldots\ldots \wedge {}_PX \\ \Lambda = \Lambda_1 \wedge \ldots\ldots \wedge \Lambda_M \\ \texttt{Decomposition} \\ \mathbf{P}(\Delta \wedge \Lambda \mid \pi) = \mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\Delta \mid \Lambda \wedge \pi) \\ \texttt{Forms} \\ \mathbf{P}(\Delta \mid \Lambda \wedge \pi) = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \Lambda \wedge \pi) \end{cases} \\ \texttt{Ident } (\delta) \end{cases} \\ \texttt{Ques} \\ \mathbf{P}(\Lambda \mid \delta \wedge \pi) \end{cases} \quad [73]$$

- δ is a collection of P data ${}_jx$. Each datum is a collection of values for the variables $X_1 \wedge \ldots\ldots \wedge X_N$. For example, a datum ${}_jx$ is given by ${}_jx = [X_1 = {}_jx_1] \wedge \ldots\ldots \wedge [X_N = {}_jx_N]$.
- Each datum, knowing the model π and the values of the parameters Λ, are considered to be independent from one another:

$$\mathbf{P}(\Delta \mid \Lambda \wedge \pi) = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \Lambda \wedge \pi) \quad [74]$$

Learning or identifying the free parameters consist in answering the question $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$: What is the most probable value of the parameters knowing a given data set δ? The answer to this question is obtained by:

$$\begin{aligned} \mathbf{P}(\Lambda \mid \delta \wedge \pi) &= \frac{\mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\delta \mid \Lambda \wedge \pi)}{\mathbf{P}(\Delta \mid \pi)} \\ &= \frac{1}{Z} \times \mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\delta \mid \Lambda \wedge \pi) \\ &= \frac{1}{Z} \times \mathbf{P}(\Lambda \mid \pi) \times \prod_{k=1}^{P} \mathbf{P}({}_kx \mid \Lambda \wedge \pi) \end{aligned} \quad [75]$$

Most often, $\mathbf{P}(\Lambda \mid \pi)$ is supposed to be uniform. Maximizing $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$ thus reduces to the maximization of the likelihood $\mathbf{P}(\delta \mid \Lambda \wedge \pi)$. However, one may very well use strong priors on the values of the parameters.

The difficulty of this problem is that the parameter space may be huge and exploring it to find the most probable value may be untractable.

5.1.2. How to compare different probabilistic models (specifications)?

Given two models π_1 and π_2, each one corresponding to a given Bayesian specification, and given some data set δ, we want to choose the model which best fits the data.

As for the search of the parameters, there is also a Bayesian formulation of this problem, by having the different models appear as different values of a variable Π:

$$\text{Prog}\begin{cases}\text{Desc}\begin{cases}\text{Spec }(\pi)\begin{cases}\text{Pertinent Variables}\\ \Delta \wedge \Pi\\ \Delta = {}_1X \wedge \ldots\ldots \wedge {}_PX\\ \text{Decomposition}\\ \mathbf{P}(\Delta \wedge \Pi \mid \pi') = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\Delta \mid \Pi \wedge \pi')\\ \text{Forms}\\ \mathbf{P}(\Delta \mid [\Pi = \pi_1] \wedge \pi') = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_1 \wedge \pi')\\ \mathbf{P}(\Delta \mid [\Pi = \pi_2] \wedge \pi') = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_2 \wedge \pi')\end{cases}\\ \text{Ident }(\delta)\end{cases}\\ \text{Ques}\\ \mathbf{P}(\Pi \mid \delta \wedge \pi)\end{cases} \qquad [76]$$

- π' is the preliminary knowledge common to both models. For instance, usually both models share at least the same pertinent variables.

The question to be answered is $\mathbf{P}(\Pi \mid \delta \wedge \pi)$: the relative probability of the 2 models. The answer to this question is given by:

$$\mathbf{P}(\Pi \mid \delta \wedge \pi) = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\delta \mid \Pi \wedge \pi') \qquad [77]$$

Comparing 2 models, one usually tries to compute their relative probabilities:

$$\begin{aligned}\frac{\mathbf{P}(\pi_1 \mid \delta \wedge \pi)}{\mathbf{P}(\pi_2 \mid \delta \wedge \pi)} &= \frac{\mathbf{P}(\pi_1 \mid \pi') \times \mathbf{P}(\delta \mid \pi_1 \wedge \pi')}{\mathbf{P}(\pi_2 \mid \pi') \times \mathbf{P}(\delta \mid \pi_2 \wedge \pi')}\\ &= \frac{\mathbf{P}(\pi_1 \mid \pi')}{\mathbf{P}(\pi_2 \mid \pi')} \times \frac{\prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_1 \wedge \pi')}{\prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_2 \wedge \pi')}\end{aligned} \qquad [78]$$

This expression is most often very easy and fast to compute.

5.1.3. How to find interesting decompositions and associated parametric forms?

Given a set of pertinent variables, given a class of possible decompositions and of possible parametric forms, and given a data set δ, we want to select the best model in the class of possible ones.

This is basically is the same problem than the preceeding one, but extended to a large number of models (all the possible models of the class). We want here to find the most probable value of Π with the probability of Π given as above by:

$$\mathbf{P}(\Pi \mid \delta \wedge \pi) = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\delta \mid \Pi \wedge \pi') \qquad [79]$$

This problem may be very difficult to solve as the number of possible models may be huge.

5.1.4. How to find the pertinent variables to model a phenomenon?

Finally, the most difficult problem of the four is to search for the pertinent variables in a given class of possible variables. It is the most difficult simply because the size of the search space is even larger than in the previous problem.

It is possible to give a bayesian formulation of this fourth problem, but this formulation is far too complicated to be presented here.

Let us now review some known algorithms and method that deal with one or several of the four preceeding questions.

5.2. Expectation - Maximization (EM)

The EM algorithm (Expectation-Maximization) (see Dempster, Laird & Rubin, 1977) is a general-purpose algorithm used to answer the first question: How to identify (learn) the value of the free parameters?

It is used in a wide variety of situations best described as *incomplete-data* problems. The idea behind

the EM algorithm is intuitive and natural and it has been formulated and applied to a variety of problems. The EM algorithm can be applied either in incomplete-data situations where data are missing, distributions are truncated, observations are censored or grouped, or in situation where the incompleteness of data is not natural or evident.

The basic idea of the EM algorithm is to associate with the given incomplete-data problem, a *complete-data* problem for which ML estimation is computationally more tractable. The complete-data problem may yield a closed-form solution to the maximum likelihood estimate (MLE). The EM algorithm consists in reformulating the problem in terms of this more easily solved complete-data problem. It establishes a relationship between the likelihoods of these two problems, and exploits the simpler MLE computation of the complete-data problem, during the M-step of the iterative computing algorithm.

Usually, the E-step consists in producing data for the complete-data problem, using the observed data set of the incomplete-data problem and the current value of the parameters, so that the simpler M-step computation can be applied to this completed data set. During the E-step, it is precisely the log-likelihood of the data which is produced, and, as it is partly based on unobservable data, it is replaced by its conditional expectation given the observed data. Starting from suitable initial parameters values, E and M steps are iteratively repeated until convergence.

5.2.1. EM and bayesian networks

In Bayesian networks (see section 3.1.1), the EM algorithm is used to answer the first question: how to identify (learn) the value of the free parameters? The searched parameters are the values of the probabilty tables associated with the vertices and edges of the network.

When some nodes are hidden, the EM algorithm is used to find a locally optimal MLE. In the E-step, we compute the expected values of hidden nodes, given observed data. This is done by using an inference algorithm (for instance JLO), and then we treat these expected values as though they were observed. The M step is carried out by using the expected values of the hidden nodes as if they were observed. The solution is to compute the frequency of the values for each node until the difference between the new distribution and the older one is less than an arbitrary threshold.

5.2.2. EM and Mixture Models

EM is the most common algorithm used to identify the parameters of Mixture Models (see section 3.1.3).

For instance, for Gaussain mixtures, it is used to find where the gaussian kernels should be set and the values of their covariance matrices.

Numerous variants of EM have been developed in the context of mixture models. A synthesis may be found in *The EM Algorithm and Extensions* by McLachlan (McLachlan & Krishnam, 1997).

5.2.3. EM and HMM: The Baum-Welch Algorithm

The learning of the models in HMM (see section 3.1.2) is performed by the Baum-Welch algorithm, a specialized version of the EM algorithm, using the maximum likelihood estimation criteria that determines the best model's parameters according to the set of data.

Intuitively, this algorithm counts the number of occurrences of each transition between the states and the number of occurrences of each observation in a given state in the training data. Each count is weighted by the probability of the alignment (state, observation).

Since many state sequences may generate a given output sequence, the probability that a HMM λ generates a sequence $\{o^0, ..., o^t\}$ is given by the sum over all state sequences (i.e, the marginal density of output sequences).

To avoid the combinatorial explosion, a recursive computation similar to the Viterbi algorithm can be used to evaluate the above sum. The forward probability $\alpha^t(i)$ is :

$$\begin{aligned}\alpha^t(i) &= \mathbf{P}(o^0 \wedge ..... \wedge o^t \wedge [S^t = i]) \\ &= \sum_{S^0 \wedge ..... \wedge S^{t-1}} \mathbf{P}(S^0 \wedge ..... \wedge S^{t-1} \wedge [S^t = i] \wedge o^0 \wedge ..... \wedge o^t) \\ &= \sum_{S^0 \wedge ..... \wedge S^{t-1}} \mathbf{P}(S^0) \times \mathbf{P}(o^0 \mid S^0) \times \prod_{i=1}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(o^i \mid S^i)]\end{aligned} \quad [80]$$

This probability represents the probability of ending at time t in state i and generating output $\{o^0, ..., o^t\}$ using all possible state sequences in between.

The Markov assumption allows the recursive computation of the forward probability as :

$$\alpha^{t+1}(j) = \sum_{i=1}^{N} [\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j])] \quad [81]$$

This computation is similar to Viterbi decoding except that a summation is used instead of a maximum.

Another useful quantity is the backward function $\beta^t(i)$, defined as the probability of starting at time t in state i and generating output $\{o^{t+1}, ..., o^T\}$, using all possible state sequences in between:

$$\beta^t(i) = \mathbf{P}(o^{t+1} \wedge ..... \wedge o^T \wedge [S^t = i]) \quad [82]$$

The Markov assumption allows the recursive computation of the backward probability as :

$$\beta^t(i) = \sum_{j=1}^{N} [\beta^{t+1}(j) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j])] \quad [83]$$

To describe the procedure for reestimation of HMM parameters, we first define $\xi^t(i, j)$ as the posterior probability that the stochastic process accomplishes the transition $[S^t = i] \rightarrow [S^{t+1} = j]$, assuming the whole sequence of observations:

$$\xi^t(i, j) = \mathbf{P}([S^t = i] \wedge [S^{t+1} = j] \mid o^0 \wedge ..... \wedge o^T) \quad [84]$$

We deduce:

$$\begin{aligned} \xi^t(i, j) &= \frac{\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)}{\mathbf{P}(o^0 \wedge ..... \wedge o^T)} \\ &= \frac{\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} [\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)]} \end{aligned} \quad [85]$$

Finally, we define $\gamma^t(i)$ as the posterior probability that the process is in the state i at time t:

$$\gamma^t(i) = \sum_{j=1}^{N} \xi^t(i, j) \quad [86]$$

Therefore, the maximum likelihood (ML) estimate of the parameters of the HMM is:

$$\begin{aligned} \mathbf{P}([S^0 = i]) &= \gamma^0(i) \\ \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) &= \frac{\sum_{t=0}^{T} \xi^t(i, j)}{\sum_{t=0}^{T} \gamma^t(i)} \\ \mathbf{P}([O^t = k] \mid [S^t = i]) &= \frac{\sum_{t=0,\, O^t = k}^{T} \gamma^t(i)}{\sum_{t=0}^{T} \gamma^t(i)} \end{aligned} \quad [87]$$

See Rabiner's paper for details (Rabiner & Juang, 1986).

5.3. Problem Oriented Models

In problem oriented models, answering the first question is usually very easy. Indeed, the variables, the decomposition and the parametric forms have been chosen with 3 main desiderata in mind:

1 Build a valid model of the observed phenomenon.

2 Build a simple enough model so that inference is kept tractable.

3 *Build a model so that the free parameters can easily be identified.*

For instance, in the sensor fusion models (see section 3.2.1), the sensor models $\mathbf{P}(S_i \mid \Phi)$ are very often Gaussian distributions. In that case, given N couples of values $\{({}_0s_i, {}_0\Phi), ..., ({}_Ns_i, {}_N\Phi)\}$, correponding to N observations, it is trivial to compute the means and standard deviations for each gaussian.

More details on this may be found in Lebeltel's Ph.D. thesis (Lebeltel, 1999) and more elaborate learning methods of this kind in Diard's Masters and Ph.D. thesis (Diard & Lebeltel, 1999 & Diard, 2003).

5.4. Learning Structure of Bayesian Networks

When learning the structure of a Bayesian network, one tries to solve question 3. Indeed, the problem is to find the best BN decomposition (i.e. the best graph structure).

Two main techniques are used to deal with this problem:

- Greedy hill-climbing: this algorithm starts from one (or several) random network and then "hill-climbs", by choosing iteratively a better network among its neighbours.
- MCMC Model Composition: This algorithm uses a Metropolis-Hastings algorithm (see section 4.3.2) to search the models using the acceptance rate:

$$a = \frac{\mathbf{P}(\pi_i \mid \delta \wedge \pi)}{\mathbf{P}(\pi^t \mid \delta \wedge \pi)} \qquad [88]$$

as in equation [50].

Starting points to this subject are Heckerman's tutorial (Heckerman, 1996) and Buntime's one (Buntime, 2002). More details on MCMC methods may be found in 2 other papers (Murphy, 2001 & Friedman & Koller, 2000).

5.5. Bayesian Evolution?

Let us finish with some long term and very exciting perspectives.

If we admit that living systems are doing Bayesian inference and learning, then there must exist some natural processes, which have been answering and are still answering our 4 questions.

1 How to identify (learn) the value of the free parameters?
2 How to compare different probabilistic models (specifications)?
3 How to find interesting decompositions and associated parametric forms?
4 How to find the pertinent variables to model a phenomenon?

A tantalising answer is to say that natural evolution provided living beings with both the pertinent variables and the adequate decomposition and parametric forms. The pertinent variables may have been obtained by selecting the sensors and actuators in order to supply vital information. The decomposition would correspond to the structure of the nervous system, which basically expresses dependencies and conditional independencies between variables. The parametric forms can be seen as the information processing units implemented by neurons and assembly of neurons.

Given this apparatus, correponding to the preliminary knowledge, each individual in its lifetime can answer the first question by experimenting and learning the values of the free parameters of its nervous system.

We plan, in BIBA, to explore this possibility, by using evolutionary techniques to answer questions 2, 3 and 4. Some preliminary work has been already done in GRAVIR on that subject.

To the best of our knowledge, only Zhang (see its publications at http://bi.snu.ac.kr/Publications/pub_ei.html#BEA) really tried to explore the same path. However, our bibliographical study on this subject is not completed yet and we may have missed important works. It will be pursued next months.

6. PERSPECTIVES

6.1. Long Term Perspectives for the Bayesian Approach

Bayesian probabilistic reasonning - more flexible and more generally applicable than classical logic - offers a new point of view that could entirely change our conception of intelligence over the coming years. The challenge of the BIBA project is to formalise and to evaluate by experiments this ongoing paradigm shift.

We hope that this state of the art makes a first step in this direction by convincing the reader of several very important points:

- There is an important and exploding interest of the scientific community in the Bayesian approaches. Important technical progress have been made in the very last years and more are coming.
- There is a common formalism and a simple theoretical core which lays sound foundations for all these approaches.
- Most importantly, probabilistic reasonning is a credible alternative to classical logic, a model of rationality when one has to deal with incomplete and uncertain knowledge, which is always the case when interacting physically with the world.

6.2. Short Term BIBA Perspectives Concerning Bayesian Programming and Modelling

One of the goal of Biba is to use the presented formalism to build biologicaly plausible models of the interactions between an autonomous system and its environment.

For us, the next step will be to help our collegue from LPPA and UCL to express their models under this formalism.

For exemple it could be possible to express the three types of learning presented in "The biological and neural handling of probabilities" (Horace Barlow and Tony Gardner-Medwin BIBA - WP1 - Oct 2002) in this formalism. Bayesian Models of gaze stabilization have allready been developped by LLPA.

They are two advantages to this approach :

1 It will help to build formal models of biological systems

2 It will eventually lead to implementation on computers thanks to the software tools which have been deveopped based on this formalism.

Also we hope to see the stream of ideas going the other way around. In particular, we would encourage our collegues to look at biological systems as machines capable of performing approximate inference (The central hypothesis of the Biba project). To do so we will try to describe the basic algorithm used in inference so that they may know where to look in biological circuit to find circuitries performing the same type of procesing

6.3. Short Term BIBA Perspectives Concerning Bayesian Reasonning

The LAPLACE research group at GRAVIR has been working during the past years on OPL: an API[6] dedicated to automate Bayesian Reasonning.

This API is made of 2 fundamental elements:

- An effective implementation of the Bayesian Programming formalism used in this paper, as a collection of C++ classes which can be used and incorporated in any program to develop applications including Bayesian computations.
- An inference engine which automates Bayesian reasonning implementing a number of techniques described in this paper, either commonly used, or more original and developed by the group.

The development of OPL will continue along the execution of the BIBA project. Different releases of this programming tool will be used by the project partners to develop their experiments.

A first version will be released with the V1 version of the robot at the beginning of year 2003.

Some early courses about OPL have already been held during the 2 BIBA schools of this first year. The next one is planned in January 2003 during the next school.

Regular updates of OPL are planned as BIBA will advance and a new release will come with versionV2 of the robot.

Of course, even if OPL is used outside the BIBA project and will soon be available as an industrial product, the OPL developers consider that the BIBA experiments and applications will be a very important test and source of inspiration for the API.

---

6. Application Programming Interface

GRAVIR hopes that important exchanges with both EPFL and LPPA will take place on this subject, as these 3 groups will work with the same robots using OPL to develop Bayesian robotic program.

6.4. Short Term BIBA Perspectives Concerning Bayesian Learning and Evolution

In the bayesian framework learning is a special case of inference, but in practice it seems difficult to use a general purpose inference engine to perform learning. Or at least the general purpose engine should be modified to accomodate a large number of variables. This modification could take two distinct forms

1 using other algorithm for approximate inference
2 using new implicite representation of decomposition

These modifications may lead to substancial modification of the current inferenc engine

7. BIBLIOGRAPHY

Aji S. M. and McEliece R. J. ; (2000) ; The Generalized Distributive Law ; *IEEE Trans. Information Theory*, Vol. 46, No. 2

Arulampalam, S., Maskell, S., Gordon, N. & Clapp, T. ; (2001) ; A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking ; *IEEE Transactions on Signal Processing* citeseer.nj.nec.com/maskell01tutorial.html

Barker, A.L., Brown, D.E. & Martin, W.N. ; (1994) ; *Bayesian estimation and the Kalman filter* ; Technical Report IPC-TR-94-002, University of Virginia

Bengio Y. & Frasconi P. ; (1995) ; An Input/Output HMM Architecture ; *Advances in Neural Information Processing Systems 7* ; Edited by **G. Tesauro and D.S. Touretzky and T.K. Leen** ; MIT Press, Cambridge, MA, USA

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K. ; (1998a) ; Interprétation ou Description (I) : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs ; *Intellectic*a ; Vol. 26-27, pp. 257-311

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K. (1998b) Interprétation ou Description (II) : Fondements mathématiques de l'approche F+D ; *Intellectica* ; Vol. 26-27, pp. 313-336

Bessière, Pierre; (2002) ; Procédé de détermination de la valeur à donner à différents paramètres d'un système ; Demande de brevet d'invention n°0235541154.4 ; Institut Européen des Brevets

Boutilier, C., Dean, T. & Hanks, S. ; (1999) ; Decision Theoretic Planning: Structural Assumptions and Computational Leverage ; Journal of Artificial Intelligence Research (JAIR), Volume 11, Page 1-94 citeseer.nj.nec.com/368281.html

Box, G.E. & Muller M.E. ; (1958) ; A note on the generation of random normal deviates ; in *Annals Math. Stat* , Vol. 29

Buntime, W. ; (1996) ; A guide to the literature on learning probabilistic networks ; *IEEE Transactions on knowledge and data engineering*, Vol. 8, P.195-210

Burgard W., Fox D., Hennig D. & Schmidt T. ; (1996) ; Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids ; Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference ; AAAI Press / MIT Press

Cacciatore Timothy W. & Nowlan Steven J. ; (1994) ; Mixtures of Controllers for Jump Linear and Non-Linear Plants ; *Advances in Neural Information Processing Systems 6* ; Edited by **Jack D. Cowan and Gerald Tesauro and Joshua Alspector** ; Morgan Kaufmann Publishers, Inc.

Cooper, G. ; (1990) ; The computational complexity of probabilistic inference using Bayesian belief networks ; *Artificial Intelligence*, Vol. 42, pp. 393-405 www-ksl.stanford.edu/KSL_Abstracts/KSL-90-34.html

Cox, R.T. (1961) ; *The algebra of probable inference* ; The John Hopkins Press, Baltimore

Dagum, P. & Luby, M. ; (1993) ; Approximate probabilistic reasoning in Bayesian belief network is NP-Hard ;

Artificial Intelligence, Vol. 60, pp. 141-153

Davis P. J. & Rabinowitz P. ; (1975) ; *Methods of Numerical Integration* ; Academic Press, New York, USA

Dean T. & Kanazawa K.; (1989) ; A model for reasoning about persistence and causation ; *Computational Intelligence*, Vol. 5, n° 3, p. 142--150

Dempster A. P., Laird N. M. & Rubin D. B.; (1977) ; Maximum likelihood from incomplete data via the EM algorithm (with discussion) ; *Journal of the Royal Statistical Society series B*, Vol. 39, p. 1-38

Diard, J. & Lebeltel, O.; (1999) ; Bayesian Learning Experiments with a Khepera Robot ; Proceedings of the first International Khepera Workshop, Paderborn, Germany, pp. 129-138

Diard, J. ; (2003) ; *La carte bayésienne -- Un modèle probabiliste hiérarchique pour la navigation en robotique mobile* ; PhD thesis, INPG, Grenoble, To be defended the 28 January 2003

Duda, R.O and Hart, P.E. ; (1973) ; *Pattern Classification and Scene Analysis* ; John Wiley & Sons, New York, USA

Erickson, G.J. & Smith, C.R. (1988a) ; *Maximum-Entropy and Bayesian methods in science and engineering ; Volume 1 : Foundations* ; Kluwer Academic Publishers

Erickson, G.J. & Smith, C.R. ; (1988b) ; *Maximum-Entropy and Bayesian methods in science and engineering ; Volume 2 : Application*s ; Kluwer Academic Publishers

Forney, G.D. ; (1973) ; The Viterbi Algorithm ; *IEEE Transactions*, Vol. 61, p. 268-278

Frey, B.J. ; (1998) ; *Graphical Models for Machine Learning and Digital Communication* ; MIT Press

Friedman, N. & Koller, D.; (2000) ; Being Bayesian about Network Structure ; Stanford University Tech. Report
citeseer.nj.nec.com/friedman00being.html

Gradshteyn I. S. & Ryzhik I. M. ; (1980) ; *Table of Integrals, Series and Products* ; Academic Press, New York, USA

Geman S. & Geman D. ; (1984) ; Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images ; IEEE transactions on pattern analysis and machine intelligence, Vol. 6

Ghahramani Zoubin ; (2001) ; An Introduction to Hidden Markov Models and Bayesian Networks ; *Journal of Pattern Recognition and Artificial Intelligence* ; Vol. 15, N° 1, p. 9-42

Halpern, J.Y. ; (1999a) ; A Counterexample to Theorems of Cox and Fine ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 10, pp. 67-85.
www.jair.org/abstracts/halpern99a.html

Halpern, J.Y. ; (1999b) ; Cox's Theorem Revisited ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 11, pp. 429-435.
www.cs.washington.edu/research/jair/abstracts/halpern99b.html

Hauskrecht M., Meuleau N., Kaelbling L.P., Dean T. & Boutilier C. ; (1998) ; Hierarchical Solution of Markov Decision Processes using Macro-actions ; Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98) ; Morgan Kaufmann

Heckerman, D.; (1995) ; A tutorial on learning with bayesian networks ; Microsoft Research Technical report
citeseer.nj.nec.com/article/heckerman95tutorial.html

Jaynes, E.T. (1982) On the rationale of maximum-entropy methods ; Proceedings of the IEEE
bayes.wustl.edu/etj/articles/rational.pdf

Jaynes, E.T. (1995) *Probability theory - The logic of science* ; unfinished book available at http://bayes.wustl.edu/etj/prob.html

Jensen, F., Lauritzen, S. & Olesen, K. ; (1990) ; Bayesian updating in recursive graphical models by local computations ; *Computational Statistical Quaterly*, 4:269-282

Jensen F. ; (1996) ; *An Introduction to Bayesian Networks* ; UCL Press

Jordan, M. I. and Jacobs R. A. ; (1994) ; Hierarchical mixtures of experts and the EM algorithm ; *Neural Computation*, 6, 181-214

Jordan, M. ; (1998) ; *Learning in Graphical Models ;* MIT Press

Jordan, M., Ghahramani, Z., Jaakkola, T.S. & Saul, L.K. ; (1999) ; An introduction to variational methods for graphical models ; *Machine Learning*

J**ordan M. I. & Weiss Y.**; (2002) ; Graphical models: Probabilistic inference ; In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks, 2nd edition*; MIT Press, Cambridge, MA, USA

Kaelbling, L.P., Littman, M.L. & Cassandra, A.R. ; (1998) ; Planning and Acting in Partially Observable Stochastic Domains ; Artificial Intelligence, Volume 101
citeseer.nj.nec.com/kaelbling95planning.html

Kalman, R.E. ; (1960) ; A new approach to linear filtering and prediction problems ; Journal of Basic Engineering, n°35
www.cs.unc.edu/~welch/media/pdf/Kalman1960.pdf

Kapur, J.N, & Kesavan, H.K. (1992) *Entropy optimization principles with applications* ; Academic Press

Lane T. & Kaelbling L.P. ; (2001) ; Toward Hierarchical Decomposition for Planning in Uncertain Environments ; Proceedings of the 2001 {IJCAI} Workshop on Planning under Uncertainty and Incomplete Information ; AAAI Press

Lauritzen, S. L. ; (1996) ; *Graphical Models* ; Oxford University Press

Lebeltel, O. ; (1999) ; *Programmation Bayésienne des Robots* ; Thèse de troisième cycle INPG, Grenoble, France.
www.inrialpes.fr/sharp/pub/laplace/publications/Rayons/Lebeltel99.pdf

Lebeltel, O., Bessière, P., Diard, J. & Mazer, E. (2000) Bayesian Robots Programming; *Les Cahiers du Laboratoire LEIBNIZ*, n°1, Mai 2000; Grenoble, France
www.inrialpes.fr/sharp/pub/laplace/publications/Rayons/Lebeltel2000.pdf

Lebeltel, O., Bessière, P., Diard, J. & Mazer, E.; (2003) ; Bayesian Robots Programming ; *Autonomous Robots* ; In press

Levine R.D. & Tribus M.; (1979) ;*The maximum entropy formalism* ; MIT Press

MacKay, D.G. ; (1996) ; Introduction to Monte Carlo Methods ; Proc. of an Erice summer school, ed. M. Jordan

McLachlan, G.J. & Krishnam, T. ; (1997) ; *The EM Algorithm and Extensions* ; Wiley, New York, USA

McLachlan G.J. & Deep D. ; (2000) ; *Finite Mixture Models* ; Wiley, New York, USA

Meila M. and Jordan M. I. ; (1996) ; Learning Fine Motion by Markov Mixtures of Experts ; In D. Touretzky, M. Mozer, and M. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press

Mekhnacha, K. ; (1999) ; *Méthodes probabilistes bayésiennes pour la prise en compte des incertitudes géométriques : Application à la CAO-robotique* ; Thèse de doctorat INPG (Institut National Polytechnique de Grenoble) ; Grenoble, France
www.inrialpes.fr/sharp/pub/laplace/publications/Rayons/Mekhnacha99.pdf

Mekhnacha K., Mazer E. & Bessière P.; (2000) ; A Robotic CAD System using a Bayesian Framework ; In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2000), Vol. 3, pp. 1597-1604, Takamatsu, Japan ; Best Paper Award.
www.inrialpes.fr/sharp/pub/laplace/publications/Rayons/mekhnacha2000a.pdf

Mekhnacha, K., Mazer, E. & Bessière, P. (2001) The design and implementation of a Bayesian CAD modeler for robotic applications; *Advanced Robotics*, Vol. 15, No. 1, pp. 45-69

Metropolis N., Rosenblusth A. W., Rosenblusth M. N., Teller A. H. & Teller E. ; (1953) ; Equation of state by fast computing machines ; *Jouranl of Chemical Physics*, Vol. 21, p. 1087-1092

Mohammad-Djafari, A.& Demoment, G. (1992) *Maximum entropy and bayesian methods* ; Kluwer Acade-

mic Publishers

Mohammad-Djafari, A. ; (2000) ; *Bayesian Inference and Maximum Entropy Methods in Science and Engineering* ; American Institute of Physics

Murphy, K. ; (2001) ; Learning Bayes net structure from sparse data sets ; Berkley University Technical report citeseer.nj.nec.com/murphy01learning.html

Murphy Kevin ; (2002) ; *Dynamic Bayesian Networks: Representation, Inference and Learning* ; PhD thesis, University of California, Berkley, USA
www.ai.mit.edu/~murphyk/Thesis/thesis.pdf

Neal Radford M. ; (1993) ; *Probabilistic inference using Markov chain Monte-Carlo Methods* ; Technical Report, CRG-TR-93-1, university of Toronto

Owen J. C.; (1993) ; Constraints on simple geometry in two and three dimensions ; *Int. J. of Computational Geometry and Applications* ; Vol. 6, n° 4, p. 421-434

Pearl, J. ; (1982) ; Reverand Bayes on inference engine: A distributed hierarchical approach ; In *Proceedings of the AAAI National Conference on A*I, Pages: 133-136

Pearl, J. (1988) *Probabilistic reasoning in intelligent systems : Networks of plausible inference* ; Morgan Kaufmann Publishers ; San Mateo, California, USA

Pineau, J. & Thrun, S. ; (2002) ; An Integrated Approach to Hierarchy and Abstraction for POMDP ; Technical Report, CMU-RI-TR-02-21, Carnegie Mellon University

Press W. H., Flannery B. P., Teukolsky S. A., Vetterling W. T. ; (1992) ; *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed* ; Cambridge University Press, Cambridge, England

Puget, Pierre ; (1989) ; *Vérification-Correction de programme pour la prise en compte des incertitudes en programmation automatique des robots* ; PhD Thesis, INPG, Grenoble, France

Rabiner L. R. & Juang B. H.; (1986) ; An Introduction to Hidden Markov Models ; *IEEE ASSP Magazine*, Vol. 3, n° 1, p. 4-16

Rabiner L.R. ; (1989) ; A tutorial on hidden Markov models and selected applications in speech recognition ; *Proc. of the IEEE Trans. on ASSP*, 77(2):257-285
www.cs.berkeley.edu/~murphyk/Bayes/rabiner.pdf

Robert, C. (1990) An entropy concentration theorem: applications in artificial intelligence and descriptive statistics ; *Journal of Applied Probabilities*

Robinson, J.A. (1965) A Machine Oriented Logic Based on the Resolution Principle ; *Jour. Assoc. Comput. Mach*, vol. 12

Robinson, J.A. (1979) *Logic : Form and Function* ; North-Holland ; New York, USA

Robinson, J.A. & Sibert, E.E. (1983a) LOGLISP : an alternative to PROLOG ; *Machine Intelligence* ; Vol. 10.

Robinson, J.A. & Sibert E.E. ; (1983b) ; LOGLISP : Motivation, design and implementation ; *Machine Intelligence* ; Vol. 10; 1983

Rubinstein R. Y. ; (1981) ; *Simulation and the Monte Carlo method* ; John Wiley and Sons

Sanderson, A.C.; (1997) ; Assemblability based on maximum likelihood configuration of tolerances ; Proc. of the IEEE Symposium on Assembly and Task Planning, Marina del Rey, CA, USA

Smith, C.R. & Grandy, W.T. Jr. (1985) *Maximum-Entropy and bayesian methods in inverse problems* ; D. Reidel Publishing Company

Smith A. F. and Roberts G. O. ; (1993) ; Bayesian computation via the Gibbs sampler and related Monte Carlo methods ; *Joural of the Royal Statistical Society B*, Vol. 55, p. 3-23

Taylor, R.H. ; (1976) ; *A synthesis of manipulator control programs from task-level specifications* ; Phd Thesis, Stanford University, CA, USA

Thrun S., Burgard W. & Fox D. ; (1998) ; A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots ; *Machine Learning and Autonomous Robots*, Vol 31, N° 5, p. 1-25

Thrun S. ; (2000) ; Probabilistic algorithms in robotics ; *AI Magazine*, Vol. 21, n° 4, p 93-109

Welch, G. & Bishop, G. ; (1997) ; An introduction to the Kalman filter www.cs.unc.edu/welch/kalman/index.html

Yedidia, J.S., Freeman, W.T., and Weiss, Y.; (2002) ; Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms; MERL corporation technical report: TR2002-35

8. ANNEX 1: NOTES ON DR EMMANUEL MAZER TRIP IN USA

8.1. Visit to Whashington University (Dieter Fox and Rajesh Rao)

Dieter Fox works in the field of robotics. He mainly works on SLAM (Simultaneous Localisation and Map Building) with particle filters. He has obtained good results building these maps with a robot equipped with a SICK and these maps can be used effectively by the robot to localize himself. He has a research project with Konolidge (SRI), Latombe (Stanford), Thrun(CMU) and Koller (Stanford). Their aim is to implement a robots pool of 100 units capable of collaborating to build maps of the environment.

He does not use a special tool box for his problems, but he believes that Hidden Markov Models methods are better than particles filters for large dimension spaces.

Rajesh Rao has just published a book "Bayesian Brain" at MIT press and proposed to organize a BIBA workshop next year at NIPS. He has also mentioned like many others the Gatsby Unit from University College of London (http://www.gatsby.ucl.ac.uk/).

8.2. Visit to Stanford University (Daphnee Koller and Uri Lerner)

Daphnee Koller is leading a large group working on Bayesian inference and learning. One of her student, Uri Lerner, works on "hybrid Bayesian networks for reasoning about complex systems" (the tittle of his thesis) .

She presented an important bibliographical review.

- Kalman filter and fast SLAM: Duran White
- Monté Calro methods: Knuth
- Exact inference: Kaerulff
- Sampling: http://www-sigproc.eng.cam.ac.uk/smc/smcpapers.html
  and http://www.cs.ubc.ca~nando/book.html
- Factorization: Murphy http://robotics.stanford.edu/~xb/ Doucet
- Dynamic Bayes Net: Russel Norvig

As in other groups the software is developed not on a documented tool box but rather above a set of in house tools:

- http://www-users.york.ac.uk/~pml1/bayes/campbell.htm
- http://www.bayes.wustl.edu/

8.3. Visit to Microsoft Research (Attias Agias and Nebojsa Jojic)

Attias Agias works on multimodal sources separation in large dimensional spaces. He is using variational methods (see Murphy & Jordan) for inference and FFT to compute sums (this should also be investigated).

At present, he works on the construction of a robot using two microphones to localize a speaker. He is using Dynamic Bayesian Networks to modelize and reduce the noise. According to him, his robot will be completely “Bayesian”.

His main interest is in ICA (Independent Component Analysis). For him, the leaders are Jutten & Herault (Grenoble).

He also mentioned Dr Mike Shadlen (http://depts.washington.edu/pbiopage/faculty/shadlen.html) from which some articles could be interesting for BIBA.

On another hand, the group developing MSBN (Micro Soft Beyesian Net) is greatly linked with text analysis but they show some interested in the work of Ronan Le Hy, PhD student in GRAVIR-Biba.

Nebojsa Jojic (http://research.microsoft.com/users/jojic/jojic.htm) has presented sources separation demonstrations. In all these demonstrations the variables number is the one of 3 images (256 x 256). The three images are used to take time into account. By this method an invisible part of the image can be restored from motion . He uses the concept audio-visual graphical model, ICASSP 2002.

8.4. Visit Nasa Ames (Cheeseman)

http://ic.arc.nasa.gov/ic/projects/bayes-group/people/cheeseman/home.html

For Peter Cheeseman, the best qualified people for Bayesian resoning in Europe are in Munich.

He works in Bayesian field but to me the methods are close from the least square method. The choice of variables taken into account are the pixels. The direct models of the camera is very detailed. That results in rendering models extremely precise. For him the problem is to use least squares with huge matrix in the order of 106. The image shifting allows to obtain a sub-pixel resolution. He also mentioned there is a problem with Maxent if all the constraints are not known.

8.5. Visit to Berkeley (Jordan)

Micheal Jordan has developed a powerful inference method (the variational method)

The method allows to effectively link an image with text. He recommended to look at the following web site to get more information about variational methods.

- http://www.merl.com/people/yedidia/
- http://www.merl.com/people/yedidia/yedidia.html
- http://www.inference.phy

TABLE OF CONTENT