Introducing contextual awareness within the state estimation process: Augmenting Bayes filters with context-dependent time-heterogeneous distributions

Alexandre Ravet

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *13/10/2015* **par :**

Alexandre Ravet

Introducing contextual awareness within the state estimation process:

Augmenting Bayes filters with context-dependent time-heterogeneous distributions

---

JURY

Alain Dutech	Rapporteur
Emmanuel Mazer	Rapporteur
David Filiat	Examinateur
Emmanuel Rachelson	Examinateur
Malik Ghallab	Examinateur

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
LAAS - CNRS
Directeur(s) de Thèse :
Simon Lacroix et *Gautier Hattenberger*
Rapporteurs :
Alain Dutech et *Emmanuel Mazer*

Introducing contextual awareness within the state estimation process:
Augmenting Bayes filters with context-dependent time-heterogeneous distributions

Alexandre Ravet

Thesis prepared at
CNRS LAAS
Robotics and InteractionS group
7, avenue du Colonel Roche, BP 52400
F-31031 Toulouse Cedex 4

with a grant from
Région Midi Pyrénées

Abstract

Prevalent approaches for endowing robots with autonomous navigation capabilities require the estimation of a system state representation based on noisy sensor information. This system state usually depicts a set of dynamic variables such as the position, velocity and orientation required for the robot to achieve a task. In robotics, and in many other contexts, research efforts on state estimation converged towards the popular Bayes filter.

The primary reason for the success of Bayes filtering is its simplicity, from the mathematical tools required by the recursive filtering equations, to the light and intuitive system representation provided by the underlying Hidden Markov Model. Recursive filtering also provides the most common and reliable method for real-time state estimation thanks to its computational efficiency. To keep low computational complexity, but also because real physical systems are not perfectly understood, and hence never faithfully represented by a model, Bayes filters usually rely on a minimum system state representation. Any unmodeled or unknown aspect of the system is then encompassed within additional noise terms.

On the other hand, autonomous navigation requires robustness and adaptation capabilities regarding changing environments. This creates the need for introducing *contextual awareness* within the filtering process. In this thesis, we specifically focus on enhancing state estimation models for dealing with context-dependent sensor performance alterations. The issue is then to establish a practical balance between computational complexity and realistic modeling of the system through the introduction of contextual information.

We investigate on achieving this balance by extending the classical Bayes filter in order to compensate for the optimistic assumptions made by modeling the system through time-homogeneous distributions, while still benefiting from the recursive filtering computational efficiency. Based on raw data provided by a set of sensors and any relevant information, we start by introducing a new context variable, while never trying to characterize a concrete context typology. Within the Bayesian framework, machine learning techniques are then used in order to automatically define a context-dependent time-heterogeneous observation distribution by introducing two additional models: a model providing observation noise predictions and a model providing observation selection rules.

The investigation also concerns the impact of the training method. In the context of Bayesian filtering, the underlying model we exploit is usually trained in the generative manner. Thus, optimal parameters are those that allow the model to explain at best the data observed in the training set. On the other hand, discriminative training can implicitly help in compensating for mismodeled aspects of the system, by optimizing the model

parameters with respect to the ultimate system performance, the estimate accuracy. In a further discussion, we also analyse how the training method changes the meaning of the model, and how this property can be properly exploited. Throughout the manuscript, results obtained with simulated and representative real data are presented and analysed.

Contents

Chapter 1

Introduction

> Remember that all models are wrong; the practical question is how wrong do they have to be to be not useful.
>
> George E. P. Box

Contents

1.1 State estimation

Estimation can be seen as the task of extracting information from noisy measurements. State estimation specifically refers to the task of estimating the state value of a dynamic system – which in robotics application might be the position, the speed, the angular velocity, etc. – by opposition to time-invariant parameter estimation. In other words, the state estimation task consists in eliminating the noise in the data providing information about the real system state, a principle depicted in Fig.1.1. This is why the estimation process is often referred to as *filtering*. More rigorously, one should refer to *optimal filtering*, since maximizing the information about the real state relies on a chosen optimality criterion. Treated as a random variable, the estimated state $\hat{x}$ is then obtained by minimizing a loss function of the estimation error $\epsilon = x - \hat{x}$, where x represents the real system state. The loss function is usually defined by the popular least squares error or the minimum mean-squared error.

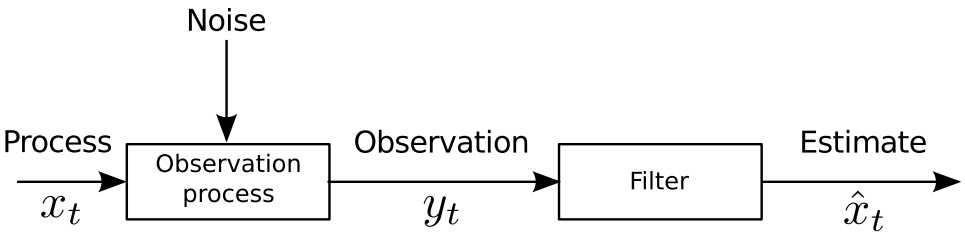


Figure 1.1: The basic state estimation process.

In this domain, the Wiener filter (Wiener, 1949) provided the first answer to the problem of filtering a stationary Gaussian process by optimizing a minimum mean-squared error criterion. This work was closely followed by the milestone article proposed by Kalman (Kalman, 1960) who extended this approach to the non-stationary and discrete-time case for a system with known linear dynamics.

While the original formulation of the Kalman filter was based on the least squares minimization method, similar equations where derived by the Bayesian community using the *Linear Dynamical System* and a purely probabilistic analysis (Jazwinski, 1970). The Kalman filter was soon discovered to be a special instance of a broader class of filters, known as the Bayesian filters[1], and a variety of new models could be treated within the unifying probabilistic framework.

Throughout the years, a diversity of Bayes filters has emerged, and have been widely used in applications such as economics, tracking, failure detec-

[1]Throughout this dissertation, we will refer to Bayesian filters or Bayes filters without distinction.

tion, control systems, demography and many others. Among all these applications, Bayes filters especially became of great importance in robotics, where their low computational cost and relatively simple implementation fostered their diffusion. In most cases, the spatial awareness required by a robot for autonomously achieving a task relies on the sole information provided by a Bayes filter. Thus, the state estimation process is of critical importance for such systems.

1.2 Model assumptions and drawbacks

Strictly speaking, the term Bayes filter (and filtering) refers to the algorithm used to perform state estimation based on a general system representation known as the *state-observation model*. This template model conceptually splits out the purely dynamical aspect of the system, considering it monitors its own evolution without any other knowledge than its dynamical equations, from the measurements made along its evolution. The graphical structure of this model is shown in Fig.1.2 and relies on two main components: a prediction distribution $p(x_t|x_{t-1})$ modeling the evolution of the system between two subsequent states, and an observation distribution $p(y_t|x_t)$ associating an expected measurement y_t to a given state value x_t. This template model gave rise to two main implementations: the Hidden Markov Model (HMM) when the state variable is discrete, and the aforementioned Linear Dynamical System (LDS) in which the state variable is continuous and both prediction and observation distributions are linear Gaussian. Note that when the state is continuous, this model is also known as the *state space model* (SSM)[2].

As an optimal technique, Bayes filtering brings the substantial benefit of making the best use of the state-observation model and the collected measurements to provide the state estimate. However, statistical optimality is no longer guaranteed if errors and approximations are made in the model.

A Bayes filter thus provides optimal results if the state variables, the joint measurements, and the causal relationships between them perfectly describe the real system. This is very unlikely for numerous real applications where the number of required state variables is too high to be tractable, or because the real physics underlying the system are unexplained. It is then very common to introduce simplifying and optimistic assumptions when describing the prediction and observation functions.

One may also consider that mapping the real process to the simple state-observation graph structure is the first structural and optimistic assumption made in the modeling stage. This is because the real causal dependencies between the state and observation variables may require a different

[2]In this thesis, we will sometimes use the term Bayes filter to generally refer to continuous state-observation models

topology. For example, the actual dynamics of the system may also depend on earlier state values, so that the prediction distribution is given by $p(x_t|x_{t-1}, x_{t-2}, ..., x_{t-T})$. In this case the prevalent *first-order* Markov chain representation has to be dropped and a more complex model emerges. Similarly, the fact that the observation y_t depends on the sole state variable is often a crude simplification of the real observation process: multiple external phenomenons affect the nature of the true observation distribution.

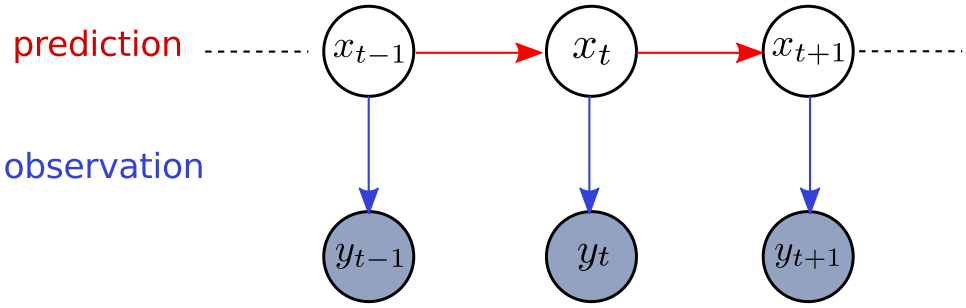


Figure 1.2: A State-space model representing the sequence of states x_t and associated measurements y_t. The transition between subsequent states is modeled by the prediction distribution, and the measurement process for each state is modeled by the observation distribution.

1.2.1 Representing uncertainty in the model

In order to compensate for the ignored or mismodeled aspects of the real system that may decrease the efficiency of the optimal filter, a description of the uncertainty in the model is then introduced. This uncertainty is represented through stochasticity (even if the system is not truly stochastic) which results in the introduction of some additional noise distributions over the deterministic component of the transition and observation models. This built-in capability to represent the uncertainty in the model itself, and ultimately in the state estimate, is a key ingredient leading to the esteemed robustness and reliability of Bayes filtering methods.

However, once again achieving optimal filtering requires an accurate representation and parametrization of the noise models. This task is complex since it consists in modeling what we actually ignore from the real system. Consequently, the noise models are often chosen for mathematical considerations, a typical illustration of this being the Gaussian noise used in the original Kalman filter. Indeed, since a Gaussian distribution whose mean is defined by another Gaussian leads to a new Gaussian distribution, this property yields the convenient closed-form expressions of Kalman recursive filtering.

As it is difficult to define a realistically accurate model, finding relevant parameters has always been mostly considered as a tuning task (Maybeck, 1982a; b). In robotics, this is commonly done by making a statistical estimation of the noise value at first, and ultimately tuning these parameters by hand once the filter is integrated in the system. In practice, the tuning stage consists in searching for the better balance between filter performance and robustness. Larger noise magnitudes allowing to encompass more unmodeled aspects at the price of a higher estimate uncertainty, while smaller values potentially resulting in overconfidence in the observation and then in inconsistent estimates (Thrun et al., 2005b). Note that intuitively, the parameter 'optimization' is then done with respect to the ultimate performance of the system, *i.e* the accuracy of the state estimate. Alternatively, parameter tuning can also be automated, or treated within the framework of state estimation by augmenting the primary state with additional variables representing the model parameters (Bar-Shalom et al., 2002). This approach however still requires manual tuning of the artificial process that models the evolution of the parameters.

Modeling uncertainty in the observation model

In the best case, a real sensor operates under nominal conditions, meaning that it provides measurements with a known characteristic noise. The deterministic observation model associated with the noise component then provides an adequate representation of the real measurement process. In real applications however, several influential and unmodeled factors cause the sensor measurements to exhibit a wide range of alterations. This means that the noise magnitude may actually vary in time, up until the measurement can not be treated as a noisy information, but should rather be treated as unreliable data. For example, it is the case for sensors being used out of their operating range, or for sensors undergoing a transient disturbance, like a vision based system exposed to low luminosity conditions. Three major issues then arise when modeling the observation noise model:

- **State estimation sub-optimality**: the noise model tries to capture multiple factors corresponding to different noise values. As a consequence, it is often impossible to find a noise magnitude that would yield an optimal estimate in any situation. Note that the noise term is not only used to capture the existence of unmodeled disturbances leading to fluctuation in the actual measurement error. The noise term also implicitly compensates for other effects like: inaccuracies in the prediction and observation models, approximation error introduced through the estimate belief representation, the discretization of time, and many others. It is thus important to realize that estimating the optimal noise value does not simply consists in making a statistical es-

timation of the measurement error. Instead, one should preferably find the optimal parameter with respect to the ultimate filter performance, hence automatically considering any aforecited effect.

- **The need for outlier rejection**: by definition outliers are measurements lying outside the nominal distribution. These outliers can totally break down the filter performance and lead to inconsistent estimates. To intuitively understand this problem, we shall remind that a Kalman filter can be viewed as a minimizer of the least-squares criterion which is known to be very sensitive to outliers. Consequently, Bayes filters often require some additional outlier rejection scheme, or the introduction of a more complex noise distribution explicitly accounting for the existence of these disturbances. These approaches respectively require to define an appropriate heuristic in the first case, and lead to a more complex implementation in the second case (Thrun et al., 2005b).

- **The exploitation Vs. rejection problem**: following the two previous points, a question naturally ensues: when can we use the measurement so that it ultimately improves the estimate accuracy, and when is a measurement considered to be destructive for the estimation process? This question arises if, as for a majority of Bayes filter, the choice is made of designing a nominal observation model augmented with a rejection scheme. In this case, the rejection rule is usually defined in a model-centric approach: the first step consists in finding the best distribution modeling the emission process for the nominal case, and all the measurements that do not fit this distribution are rejected. This simple mechanism relies on an artificial rejection threshold over the measurement noise which may require hand-tuning. This approach can be legitimately questioned since the rejection decision is based on the fact that a measurement does not fit a model which is known to be wrong. In other words, these typical implementations ignore the only impartial criterion for rejection, *i.e.* can a measurement help in improving the estimate accuracy in absolute terms ?

The aforementioned issues mainly concern the specific instance of Bayes filters associated with a measurement rejection rule. Other approaches exist, in which a complex observation model tries to embed the whole range of measurement alterations within the same distribution (Thrun, 2001; Thrun et al., 2005a). As described in the next subsection, there still remains a major issue that prevents us from giving the appropriate treatment to the measurements.

The closed-world assumption

Designing a state-observation model is usually done under an overall assumption: the conventional idea of eventually defining a closed-world model. A closed-world model assumes that the model (and by consequence, the state estimate) contains everything the estimation system needs to know. This prevailing assumption is inherited from a simple modeling paradigm: to obtain good results, we simply have to define a good model. However we have already seen that real systems can not always be perfectly and faithfully modeled. And if accommodating the whole model uncertainty within a noise distribution allows to embed knowledge about the existence of external factors, we are still *blind* to the occurrence of such effects at runtime. Therefore, we have no guarantee to give the appropriate treatment to a new observation, especially if small errors were made at any level in the model. Note that this is true whether we defined a complex observation distribution embedding the existence of multiple alterations or if we defined a nominal observation distribution associated with a rejection scheme. For these reasons, a filter may momentarily converge to an estimate value that makes sense for the model, but which is actually inconsistent regarding the real system. In some extreme cases, the filter may never recover from this perturbation, leading to *filter divergence*.

Conclusion

To summarize, probabilistic models provide the strong advantage of encoding a description of the uncertainty in the model due to the inevitable approximation of the real system. This makes probabilistic models considered to be very flexible in comparison to a model trying to describe precisely any possible situation. Unfortunately this flexibility can also turn into a weakness since real implementations tend to sweep numerous mismodeled aspects under the carpet that is the stochastic component of the model. Thus the uncertainty model might be overused, leading to sub-optimal estimation in the best case, and to divergence in the worst case.

1.3 Enhancing Bayes filters for a better balance between modeling errors and time efficiency

As the research context motivating this thesis concerns at first the issue of robust and adaptive estimation for autonomous robots, the following discussion focuses only on the observation component. However the ideas proposed along this manuscript can be straightforwardly applied to the prediction distribution.

Why improving a structurally limited model ?

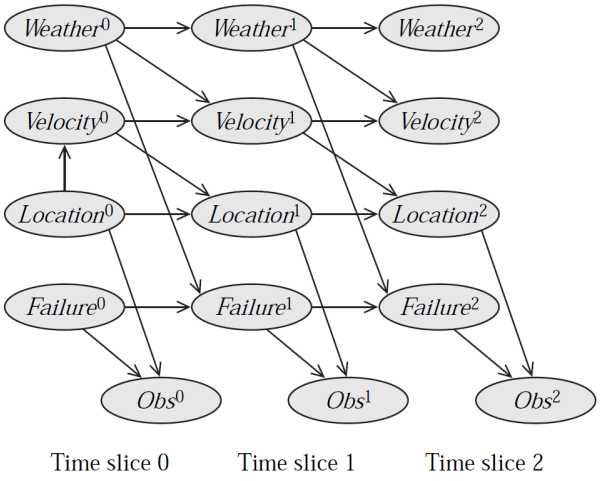
It is clear that the basic state-observation model depicted in Fig.1.2 inherently ignores some aspects of the underlying complexity of a real system. Especially, the simplistic topology of the state-observation model can be seen as the main weakness causing the previously mentioned issues when modeling the complex measurement generation process. One might be tempted to overcome this limitation with a simple answer: a complex process requires a complex model. In other words, the observation component should not be represented by a single state $\rightarrow$ observation link, but should involve additional variables and causal relationships to express any phenomenon involved in the measurement process. This is why the state-observation representation can be put in perspective with temporal models capable of encoding additional causal dependencies like the general Dynamic Bayesian Networks (DBN) (Dean and Kanazawa, 1989). Additional modeling power can also emerge from the introduction of complex features defined over the measurements. This approach fostered the development of the Conditional Random Fields (CRF) (Lafferty, 2001), which also presents inherent capabilities to easily expand the topological structure of the graph. A graphical illustration of a simple DBN, and the *linear-chain CRF* designed for modeling the temporal aspect of a system are shown in Fig.1.3. Note that the state-observation model can be equally seen as a simple instance of DBN known as the 2 time-slice Bayesian Network (Koller and Friedman, 2009).

Defining more complex networks however requires additional domain expertise. In some cases, we may not have a sufficient knowledge about the system to define properly the required system variables. The measurement process may also be so complex that we may have to turn to structure learning methods in order to define the causal dependencies within the model. The resulting increased complexity also comes at an additional computational cost, and hence a lower estimation frequency. Besides, and as will be discussed later, increasing the model complexity does not always guarantee better results, and care has to be taken when learning the model parameters to ensure appropriate generalization when confronting the system with new data.

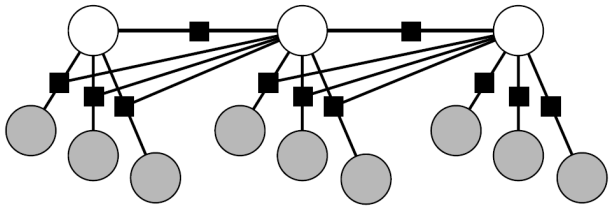
Two major conclusions naturally follow the issues discussed so far:

- The variables actually involved in the measurement generation process are not always known, neither are their probabilistic interactions. Moreover a realistic model might not always be tractable. Thus, approximations are always made somewhere, whatever the complexity of the model.

- Modeling approximations being compensated by the stochastic component of the model, a better knowledge about how the noise com-



(a) Toy example of a DBN for monitoring a vehicle. The model explicitly introduces a failure hidden state that impacts the observation process. (Image excerpt from (Koller and Friedman, 2009) p.203)



(b) Linear-chain CRF with observation-dependent transition. Following the factor graph representation, circles represent variables nodes, and the shaded squares represent factor nodes where each factor is defined as a function of its direct neighbour variables. These factors are used to compute the conditional distribution of the hidden variables given the observations. Note that the observation generation process is not modeled in a CRF, and the observations are only exploited through the factor nodes. This model property allows for the straightforward introduction of arbitrarily complex features. (Illustration excerpt from (Sutton and McCallum, 2007) p.9)

Figure 1.3: Two models extending the modeling complexity of the state-observation model.

ponent can make the model more faithful to reality, and embedding this knowledge within the model, would help in getting closer to the optimal estimate. That is, one should allow the model to estimate the actual noise magnitude instead of encompassing multiple situations within a single static noise value.

The approaches presented in this manuscript stand on a simple paradigm: excellent results can theoretically be obtained with a simple model, as long as we continuously know to which extent its modeling assumptions are faithful to reality. Since approximations always have to be made (regardless of the model complexity), and are further compensated by stochasticity, we foster the use of simple structures, hence decreasing the amount of knowledge required to derive a proper noise component.

Besides these considerations, we should remind that the state-observation model provides a natural and unifying notation which proved to be highly robust to its own approximations, and is still the backbone of temporal processes modeling. One recent example being the linear-chain CRF which can be seen as the discriminative analog of the HMM (Sutton and McCallum, 2007).

For these reasons, we investigate on approaches for improving the state-observation model reliability by compensating for its inherent mismodeled aspects, aiming for a practical balance between complexity and time efficiency (and implementation ease as well). Three key ideas are used along this thesis for getting better, closer to optimal estimates while using the state-observation model as a core structure:

- Introduce time-varying models.
- Use additional contextual information within the estimation process.
- Use learning algorithms to find the model parameters.

Each of these points are developed hereafter.

The need for time-varying models

As explained previously, the first reason for sub-optimal estimation is due to the classical representation of the uncertainty in the model through a single static noise component. This is because the state-observation model was built upon the *first-order Markov chain* theoretical framework which was described at first under the assumption of time-homogeneity. This assumption simplifying the mathematical manipulation and the exploitation of such models. According to this assumption, the parameters of the transition and observation distributions do not vary in time.

However, real systems are indubitably time-heterogeneous, since for example, the factors affecting the measurement generation process vary in time with the different contexts. One first improvement on the state-observation model then consists in relaxing the constraint of homogeneity and embed time-heterogeneous distributions. Following our modeling paradigm, and keeping the state-observation system representation, we do not focus on explaining a real, complex measurement process, and instead aim at smartly re-evaluating at each filter iteration the noise magnitude within the observation distribution $p(y_t|x_t)$.

Similarly, making smart decisions about measurement rejection (or analogously selection) implies adapting the decision rule over time, by evaluating the reliability of a measurement and the information it brings to the estimation process at each time step.

The need for contextual awareness

Whether it concerns noise adaptation or measurement rejection, time varying rules imply to transpose the knowledge traditionally encoded within the model itself to a distinct component providing at runtime the proper information to the state-observation structure. Also, we want this additional component to take as input the current information known to be impacting the measurement process, that we call the contextual information.

When addressing the problem of state estimation for autonomous robots, the nature of the navigation context is, indeed, of major importance in the measurement generation process. Luminosity conditions might impact the performance of cameras and laser based sensors. Some sensors may have very specific operating range, or the actual speed of the robot and roughness of the terrain might change the precision of the inertial measurements.

Through this dissertation, the notion of context encompasses without any distinction every factor that is known to impact the reliability of a sensor measurement, by comparison to its nominal operating mode.

To illustrate our approach, let's take the example of a GPS receiver. The position provided by the GPS can exhibit important drifts caused by multi-path reflections, a phenomenon more likely to happen in urban environments than in open environments. As said before, it is hard to represent this complex phenomenon through the introduction of new variables that would provide a better – and hypothetically perfect – observation model. However, we can exploit a more general information: the fact that there exists a relationship between the nature of the environment and the decreased accuracy of the measurements.

It might be tempting to include the required context variables in the state representation of the system. This approach would therefore closely

follow the approach suggested in (Bar-Shalom et al., 2002) to incorporate the model parameters within the state. This would require a description of the underlying transition and observation processes, that is highly complex. It is even debatable if this precise information can straightforwardly be modeled by a state-observation model since by nature the context is not a real dynamical process. An other side-effect emerges from this approach: by describing the context as part of the state, we incur the closed-world modeling issues previously mentioned. The consequences in this case are even worse since errors made in the model may lead to poor estimation of the filter parameters, which in turn will decrease the global quality of the estimate, meaning that the system is self-feeding with errors.

Therefore, we suggest an approach where the contextual information is not tracked by the estimation system, but is preferably captured directly by a dedicated set of sensors and any available information source. This means that the context is seen as a set of raw measurements, and that we never try to explicitly extract (or filter) its real nature. Note that this task would rather correspond to *place classification*, and it is not our goal to extract any meaningful categorization of the context. Our approach is more data-centric in that we do not expect to understand or identify what information makes sense for the robot in order to evaluate the reliability of its measurements.

Learning is the key

Once the nature of the distributions encoded by the model is fixed, determining the distributions parameters is of critical importance for the resulting quality of the state estimate. As explained before, this task has been long seen as a tuning task, and the use of learning algorithms in this domain is surprisingly recent. By automatically extracting parameters from accumulated data, learning not only decreases engineering costs, but also the need for expert knowledge. It provides reliable and efficient models that proved to outperform the ones built after human expertise (Abbeel et al., 2005). In fact, learning, as a data-driven approach, proved to be very effective in many domains, and not only in Bayesian filtering. As explained in (Koller and Friedman, 2009), this is because "*The models produced by this process are usually much better reflections of the domain than the models that are purely hand-constructed*".

For our problem especially, learning the parameters turns to be the most suitable approach. At first, we increase the number of parameters to be determined and hence the engineering cost, by introducing a new probabilistic relationship between the context variables and the measurement reliability. In some cases, and as will be described later, the number of new parameters may simply be to high to be tuned through any non-automatized process. The main reason for learning the parameters however, is that we are unable to understand the real phenomenon underlying this new causal relationship.

Thus there is no effective methodology for estimating its parameters, neither for tuning them with the help of human expertise.

Two of our major concerns, automatic parameter estimation and time-heterogeneity, are inherently connected. In the domain of *adaptive filtering*, time-heterogeneity is a direct consequence of the continuous estimation of the parameters. As such, these methods appear as an appealing solution to our problem since they aim at adjusting the noise magnitude in time without requiring a precise description of the phenomenon involved. In fact, the state-observation structure does not have to be augmented since the noise level is assessed directly from data. Following the conventional Bayesian approach, online estimation of the model parameters is done by computing their posterior distribution based on the Bayes rule:

$$p(\Theta|y_{1:T}) \propto p(y_{1:T}|\Theta)p(\Theta)$$

where Θ is the vector set of parameters and $y_{1:T}$ the measurement collected up to time T. The parameters estimation then rely on two components: the prior distribution $p(\Theta)$ and the likelihood $p(y_{1:T}|\Theta)$. This later distribution basically represents how likely is the set of parameters Θ to explain the measurements collected, according to the state-observation model. For this purpose, these methods implicitly assume that the state estimate provided by the filter itself stays consistent with the true state, and that it is then possible to statistically estimate some of the model parameters based on this information (Bar-Shalom et al., 2002). These methods inherently suffers from the closed-world representation issue, as once again we only rely on the knowledge represented by the state-observation model to explain some of its own parameters.

We have already seen that additional information (the context) is required if we want to properly assess the reliability of measurements in time, and avoid to rely on the closed-world assumption. If we want to elude the ambiguities in the estimation process naturally arising from the altered measurements, we must however fulfil an other requirement: being given either all or a subset of the true state variables, *i.e* a ground truth. To us, the ground truth provides the only sound information for either assessing a consistent noise model, or properly evaluating the reliability of a measurement. This allows to relax some of the assumptions made when learning the model, such as the hypothesis of state consistency, or assuming that the more sensors provide similar information, the more reliable they are. Note that these later approaches have been widely developed within the field of *model selection* (Burnham and Anderson, 2002) or *fault detection*.

The true state value being clearly inaccessible during normal operation, we determine the parameters of our model in an initial training phase. This requires to define some training sets containing the value of the true state and the associated measurements provided by the sensors during system

operation. This approach is referred to as *supervised learning*[3], and consequently comes at the price of a highly accurate instrument providing the ground truth. Still, knowing the true state value brings an other interesting advantage: the ability to train the model with respect to the ultimate filter performance. By optimizing the model parameters so that the output estimate is close as possible to the ground truth, we implicitly handle the modeling approximations mentioned in 1.2, and thus improve the robustness of the estimation system.

1.4 Contributions

The main contributions of this thesis are:

- Improving the classical Bayesian filtering framework limitations through the introduction of new noise adaptation and measurement selection techniques.
- Restraining the closed world assumption usually made in the filter by introducing contextual information within the estimation process.
- Suggesting the use of supervised learning and non-parametric models as an answer to the complex problem that is modeling the effect of the context over the observation generation process.

The theoretical developments provided in this manuscript follow a purely Bayesian treatment. Although it is originally motivated by robotics applications, this work can be seen as a generic improvement of the classical Bayes filter implementations. Thus, most of the following analyses and developments are not domain-specific.

1.5 Outline

The following chapter (Chapter 2) introduces the reader to the probabilistic graphical models framework. This framework forms a powerful tool for analysing and formalising our problems through an intuitive diagrammatic representation. We especially believe that this representation is a key ingredient in developing new probabilistic models, and constantly try to put our work in context following this framework.

Based on these theoretical concepts, Chapter 3 starts with a detailed description of the state observation model. Following a purely Bayesian approach, we see how inference on the model lead to the well known recursive filtering equations, and examine the usual training methods. We

[3] As opposed to unsupervised learning, which addresses the problem of data clustering and density estimation.

then discuss the core issues resulting from the original assumptions made in the state-observation model, but also from the training method. We consequently suggest a new modeling paradigm allowing for a better robustness regarding context influence over the observation. This new model results in two sub-problems, namely measurement selection and noise adaptation, that are tackled in the following chapters. At the same time, we evaluate the consequences of discriminative training regarding dependencies encoded in the model, and provide a new discriminative interpretation of the state observation model.

In Chapter 4 and 5, we address the problem of context-dependent measurement selection through two distinct, albeit related approaches. The basic idea is to define precise activation regions for each measurement subset in the context input space. Then, based on this decision boundaries, the different measurement are selected at runtime following different mixing strategies. One innovation underlying these approaches is the exploitation of the multiple model technique for measurement selection. An other important aspect is the relaxation of the concept of outlier, in that the measurement utility is not seen as a measure of its vicinity to a prior distribution. In our system, measurements are preferably selected according to their capacity in improving the ultimate state belief.

The first solution partly relies on the Mixture of Experts framework which provides an elegant treatment of the selection task within a single unifying model. The next approach tries to enhance the modeling complexity of the decision boundaries while still providing fast inference capabilities. This balance is reached thanks to the powerful Relevance Vector Machine model, which is detailed in Chapter 5. A clear understanding of this model being necessary for the following chapter, we provide a sound theoretical background before discussing its application to the measurement selection task. We provide illustration for both approaches on simulated and real data in the context of altitude estimation.

Chapter 6 tackles the measurement noise adaptation task which can be seen as a continuous counterpart of the selection task. Still constrained by the computational efficiency required for online state estimation, we exploit the regression form of the Relevance Vector Machine for modeling the noise component. The strong modifications within the state observation model significantly impacts the inference task whether it concerns the evaluation of the noise value itself, or more importantly the recursive equations. We then determine a set of approximations for simplifying the exploitation of the model, and alternatively provide solutions for generative and discriminative training.

Finally, Chapter 7 summarizes the work presented and discuss mid-term and long-term future work directions.

Chapter 2

Probabilistic graphical models

Contents

Two frameworks are pervasive in this manuscript: the *probabilistic framework*, and the *probabilistic graphical models*. The probabilistic framework forms a powerful and unifying tool for addressing many problems from a new perspective, the probabilistic treatment of the original Kalman filter mentioned in chapter 1 being an interesting illustration of this ability. One of the most interesting concept provided by the probabilistic framework is its inherent capability to handle and quantify uncertainty. This concept is particularly central in our work, as uncertainty is unavoidably brought by noise in the measurements, and because our ability to model the system is very limited, hence uncertain. In the case of Bayesian filtering, the ability to assess a confidence level about the state estimate is also fundamental in order to get consistent results. One other aspect pleading for the need to consider uncertainty is the finite size of the training sets. Indeed, even if the data-driven approach can be very effective, it is still important to be able to evaluate the degree of confidence we have in the knowledge we extracted from a limited amount of data.

Logically, the probabilistic approach handles the uncertainty in the system state by reasoning exhaustively about the multiple possible state values, and by evaluating the probability of each hypothesis being true. A central foundation behind this ability is the exploitation of the *Bayes theorem*, allowing to compute the contrapositive conditional distribution of an event a given an event b through:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

The probabilistic analysis of a system has been early supported by the use of graphical structures for representing the interactions between variables. For example this idea was proposed in 1902 by J. W. Gibbs in the domain of statistical physics. From these early developments the contemporary Probabilistic Graphical Model (PGM) framework inherited many terms and concepts. In the field of Artificial Intelligence however, it is only in the late 1980s that the probabilistic methods in association with the graphical models finally got widely adopted. This success being driven by its consistent formalism, and its sound theoretical foundations that overtook alternative approaches for reasoning under uncertainty (Shafer and Pearl, 1990). One particularly central topic within the PGMs in our work is the Bayesian network framework and the joint concept of Bayesian inference developed by Judea Pearl in (Pearl, 1988).

While PGMs are now used in a wide range of domains where they led to various implementations and algorithms, they invariably rely on three key components that define, at the same time, a complete methodology for designing a new model and reasoning under it.

These three components are:

- System representation
- Inference
- Training

In the next sections we give a proper description of each of these concepts and especially focus on the Bayesian Networks and their temporal instantiation, the Dynamic Bayesian Networks that are essential in this work. The framework of PGMs being a vast topic, many aspects are voluntarily omitted in this chapter, especially concerning the inference methods. For an exhaustive and rigorous discussion about the PGMs and probabilistic approaches, we encourage the reader to refer to the book of Daphne Koller and Nir Friedman *Probabilistic Graphical Models: principles and techniques* (Koller and Friedman, 2009).

2.1 System representation

The PGM framework, in association with probabilistic reasoning, originates from a fundamental and simple idea. That is, the idea of separating the knowledge about the system from the reasoning algorithm that exploits this knowledge to provide answers. This concept is known as *declarative representation*. In this section we describe the notions and semantics specific to the representation of knowledge, *i.e* the model of the system.

2.1.1 Bayesian Networks

The Bayesian network representation is based on directed graphs in which the nodes are the random variables of the system and the directed edges intuitively correspond to the causal influence of a node on another one. An equivalent interpretation of this graph is that it provides a representation of the independence assumptions made in the model, then described by the absence of edges. More rigorously, these graphs are referred to as *directed acyclic graphs* (DAGs), as they present two main characteristics. At first, a pair of nodes X_i, X_j is always connected by a directed edge $X_i \mapsto X_j$, by opposition to an undirected edge $X_i - X_j$. Note that undirected edges are used in an other graphical representation, namely the *Markov Networks*, also referred to as *Markov Random Fields*. The second important characteristic is that they contain no directed path $X_1 \dots X_K$ forming a cycle such that $X_1 = X_K$. This restriction is due to the difficulty of defining a coherent and practical probabilistic model when directed cycles are present, as convergence issues arise in inference and training.

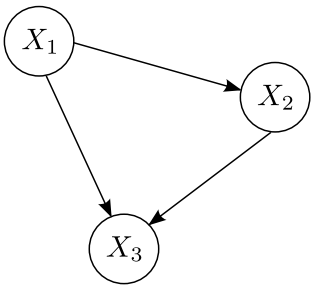


Figure 2.1: A DAG representing the joint probability distribution over three variables X_1, X_2, X_3, according to the specific decomposition described in 2.2

Hence, a DAG $\mathcal{G}$ provides an intuitive and compact representation of the joint probability distributions over the variables $X_1, \ldots, X_n$ describing our system. To illustrate how a Bayesian Network encodes knowledge about the dependencies (or independencies) between a set of random variables, we have to introduce the notion of *factorization*. For a graph $\mathcal{G}$ with n nodes, a distribution P over the variables $X_1, .., X_n$ factorizes according to $\mathcal{G}$ if $P(X_1, .., X_n)$ satisfies:

$$P(X_1, .., X_n) = \prod_{i=1}^{n} P(X_i | Pa_{X_i}^{\mathcal{G}}) \tag{2.1}$$

where $Pa_{X_i}^{\mathcal{G}}$ denotes the parent of X_i in $\mathcal{G}$.

Let us take the example of a system containing three random variables X_1, X_2, X_3, and decompose the joint distribution $P(X_1, X_2, X_3)$ by successively applying the product rule:

$$\begin{aligned} P(X_1, X_2, X_3) &= P(X_3 | X_1, X_2) P(X_1, X_2) \\ &= P(X_3 | X_1, X_2) P(X_2 | X_1) P(X_1) \end{aligned} \tag{2.2}$$

This specific decomposition is called the chain rule for the Bayesian networks. It corresponds to the Bayesian Network depicted in Fig.2.1 where each conditional distribution is represented by one, or multiple edges. Interestingly, a different decomposition of $P(X_1, X_2, X_3)$ would lead to a different network, so that the conditional dependencies made in (2.2) only hold for the graph described in Fig.2.1.

The factorization property leads to the rigorous definition of Bayesian Networks:

Definition. A Bayesian Network is a pair $(\mathcal{G}, P)$ where P factorizes over $\mathcal{G}$, and where P is specified as a set of conditional probability distributions associated with $\mathcal{G}$'s nodes.

Thus, the first step in defining a model consists in choosing a set of random variables that describe some relevant aspects of the system. This step is critical as the variables we decide to use substantially impact inference and learning, and thus the global model performance. There are two kind of variables in a model: the variables we can potentially observe, and the ones we can not observe, referred to as *hidden variables*. This later category being relevant when we know that there exists a variable in the system that we can not observe, but that is too important to be neglected. Sometimes, introducing a hidden variable of no direct interest for our application can also improve the general performance of a model as it generally increases its modeling power and thus allows to take into account some latent phenomena in the system. Care has to be taken in order to chose a sufficient amount of variables and, if possible, one has to evaluate their utility as for example, including a variable that is known to impact an other one but that can not be observed does not bring more information. In fact, these different hypotheses are already encompassed by the probabilistic representation of each event. Clearly, one has to keep in mind that all variables are not useful for a specific application, and that a suitable set of variables depend on the information we want to extract from the model.

The interpretation of the conditional distributions in the graph then depends on the nature of the variables, and on our knowledge about the causal relationships between them. Note that no information about the conditional distribution itself is provided by the graph, except when the parameters of the distribution themselves are seen as random variables. The nature of the conditional distribution is also chosen for practical reasons. For instance, a prevailing approach consists in choosing the conditional distribution so that both the prior distribution $P(Pa_{X_i}^{\mathcal{G}})$ and the posterior distribution over X_i, that is $P(X_i|Pa_{X_i}^{\mathcal{G}})$ for a specific instance of $Pa_{X_i}^{\mathcal{G}}$, have the same functional form. In that case, the prior and the conditional distribution $P(X_i|Pa_{X_i}^{\mathcal{G}})$ are said to be *conjugate* distributions, and information can be easily propagated between a parent and a child since the posterior is given by a closed-form expression[1]. The notion of conjugate distributions plays a very important role, as a graph that uniformly satisfies this property can therefore be arbitrarily extended without changing its probabilistic analysis. This property automatically holds for discrete variables, and is also satisfied in the continuous case when both the parent and the child distributions are in the popular exponential family. A very common illustration of the continuous case is the use of Gaussian distributions, which is of central interest in our work.

When the variables within the model are discrete, one has to turn to *tabular conditional probability distributions*. These tables provide an exhaustive description of the probability for a variable to take one of its possible values

[1]Note that this is actually true for propagating information both ways.

given the value of its parents. The basic tabular representation however requires a number of parameters (the table fields) that increase quickly with the graph complexity. Alternative and compact representations are then commonly used, based on some knowledge we have about deterministic dependencies, some context-independent regularities in the parameters, but also by introducing general parametric models that do not require to enumerate each conditional probability value. One popular illustration of this later approach is the sigmoid model used in logistic regression.

When the variables take value in a continuous space, many different distributions can be used for representing a conditional distribution $P(X_i|Pa^{\mathcal{G}}_{X_i})$. Defining an adequate conditional distribution usually means finding the proper parameters of a known distribution, like the mean and variance of a Gaussian for example. We will also see that some powerful methods allow us to make no assumption about the underlying functional form of the actual distribution, but rather extract a model of the distribution directly from data. Note that defining a proper conditional distribution $P(X_i|Pa^{\mathcal{G}}_{X_i})$ is equivalent to the regression task, especially when $Pa^{\mathcal{G}}_{X_i}$ can be observed.

To conclude, the probabilistic framework does not put any restriction over the nature of the variables in a graph, so that purely discrete, purely continuous, and hybrid networks can be treated with similar methods. Examples of some conditional distributions will be provided in this chapter and all along the manuscript. If defining the conditional dependencies within the graph is a core issue in obtaining an efficient model, the independencies that are implicitly defined are also very important. Especially, the concept of conditional independence is crucial for the exploitation of the model. In the next subsection, we discuss how a graph also encodes the conditional independence property.

2.1.2 Conditional independence

As seen before, the joint distribution over the system variables is decomposed into a product of conditional distributions that relate the dependencies within the model. However, the decomposition of the joint distribution $P(X_1, .., X_n)$ also encodes an important property, namely the conditional independences that exist in the model.

From a purely probabilistic point of view, a variable X_1 is said to be conditionally independent from an other variable X_2 given X_3 if we can write

$$P(X_1, X_2|X_3) = P(X_1|X_3)P(X_2|X_3)$$

In other words, where the basic product rule always allows us to write $P(X_1, X_2|X_3) = P(X_1|X_2, X_3)P(X_2|X_3)$, we now have $P(X_1|X_2, X_3) = P(X_1|X_3)$, meaning that if X_3 is given (or observed), then X_1 is statistically independent from X_2.

The graphical representation of the system also encodes the property of conditional independence. This key feature leads to an important simplification of the inference and learning task under that graph. To see how conditional independence is represented in a graphical model $\mathcal{G}$, we give at first three examples of directed graphs containing three variables and describe how an observation can lead to conditional independence. Based on these examples, we then give the more general definition of conditional independence in a graph, known as *d-separation* (Pearl, 1986).

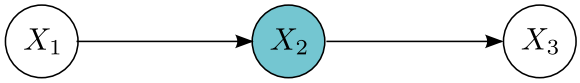
EVIDENTIAL TRAIL: Having a chain $X_1 \rightarrow X_2 \rightarrow X_3$, we intuitively understand that information can flow from X_1 to X_3 if X_2 is not observed. However, if we are given an observation of X_2, then the trail is said to be inactive as X_1 does not give additional information about X_3. This situation is described in Fig.2.2a, where a shaded node indicates that the associated random variable is observed. Consequently, for this configuration we have that X_1 and X_3 are conditionally independent given X_2.

COMMON CAUSE: For a chain $X_1 \leftarrow X_2 \rightarrow X_3$ as described in Fig.2.2b, X_2 is a common cause for both X_1 and X_3. This means that if we do not know X_2, then by probabilistically reasoning, and assuming that X_1 is in one particular state, this gives us information about the probable state of X_2. Based on that information, that means that we can infer in turn a probable value for X_3 given a particular assumption about the value of X_1. In other words X_1 and X_3 are correlated and the trail is active. However, this information flow is broken when X_2 is observed, since making an assumption about X_1 does not give additional information about X_3 anymore. Here again, X_1 and X_3 are conditionally independent given X_2.

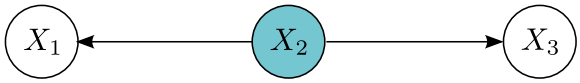
COMMON EFFECT: The last case is a structure $X_1 \rightarrow X_2 \leftarrow X_3$ as described in Fig.2.2c. This scenario is different from the two previous one in that X_2 is influenced by both X_1 and X_3. Then, assuming X_1 is in a specific state gives us information about the possible states of X_2, but all hypotheses about X_3 are still to be considered equally. In fact, it is only when X_2 is observed that, by assuming a particular value about X_1, we can probabilistically assess the value of X_3. In that case, X_1 and X_3 are conditionally independent only when X_2 is not observed.

For the general case of complex directed graphs where there can be more than one trail between two nodes, we have to consider all the possible paths. Independence is then ensured by the property of *d-separation*[2] which is defined as follows:

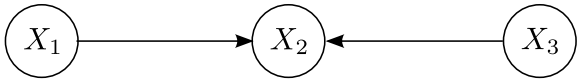
[2]The term d-separation stands for directed graph separation.



(a) The evidential trail structure: conditional independence between X_1 and X_3 holds when X_2 is observed (indicated by a shaded node).



(b) The common cause trail structure: conditional independence between X_1 and X_3 holds when X_2 is observed.



(c) The common effect trail structure: X_1 and X_3 are statistically independent when X_2 is not observed.

Figure 2.2: Three graph structures yielding independence between X_1 and X_3.

Definition. If X_1, X_2 and X_3 are three set of nodes in a graph $\mathcal{G}$, X_1 and X_3 are said to be d-separated given X_2 if there is no active trail between any node $x_1 \in X_1$ and $x_3 \in X_3$ given X_2.

In fact, a graph $\mathcal{G}$ defines a whole group of joint probability distributions as, for instance, discrete and continuous distributions can be implied by the same graph[3]. It can be shown that this group of distributions can equivalently be obtained by selecting the distributions that factorize over $\mathcal{G}$ according to 2.1, or by selecting the distributions that satisfy the conditional independencies resulting from the d-separation property in $\mathcal{G}$. In other words, a distribution P factorizes over $\mathcal{G}$ if and only if P satisfies the local independences of $\mathcal{G}$ derived from d-separation (Koller and Friedman, 2009).

Consequently, if one intuitively sees a graph structure as a visual mapping of the dependence assertions between the variables of a system, great attention should be paid regarding the independencies that implicitly emerge from this purely causal analysis. Indeed, it is very important to understand clearly how the resulting model reflects our interpretation of the system, and especially when we can consider that two variables are not statistically

[3]Also, different functional forms can be chosen for describing the conditional distributions.

dependent. Further, the independence assumptions do not only change the meaning of the model. In fact, the d-separation property can also lead to great simplifications throughout the graph exploitation. Nevertheless, if the computational tractability of a model is a main concern, one should however keep in mind that introducing conditional independence can potentially decrease the trustworthiness of the model. In the next subsection we give examples of simple Bayesian networks, and discuss how the d-separation property influences the exploitation of the graph.

2.1.3 Graph examples

Illustration: The Naive Bayes model

One of the simplest Bayesian network is the so called naive Bayes model. Its simplicity, and the strong independence assumptions made in this model makes it to be also referred to as the simple Bayes or the Idiot Bayes model.

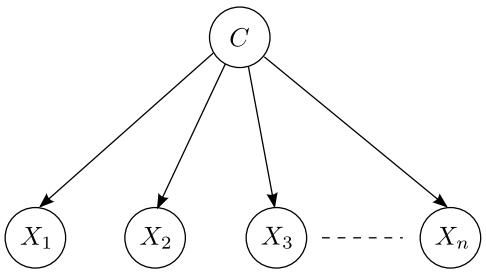


Figure 2.3: The graphical structure of the naive Bayes model for a hidden class C and n features.

The naive Bayes model is originally used for determining the class of a single variable C based on a set of observed features $X_1, \ldots, X_n$. In the text categorization application for example, C corresponds to the category of a document, for instance history, science, politics, etc, and the features X_i are often defined as word frequencies. The Bayesian network graph of this model is represented Fig.2.3 and depicts each observed variable as a direct child of the class variable. This specific structure leads to the well known naive Bayes assumption: all observed variables are conditionally independent for a given class instance. Clearly, this is because any triplet $\{X_i, C, X_{i\neq j}\}$ forms a distinct common cause trail which is inactive when C is observed.

Hence, the joint probability distribution $P(C, X_1, \ldots, X_n)$ factorizes in a simple product:

$$P(C, X_1, \ldots, X_n) = p(C) \prod_{i=1}^{n} P(X_i|C) \tag{2.3}$$

Even if it relies on a strong assumption, *i.e* all features are independent given a class, the naive Bayes model brings interesting properties due to the algebraic separation of each feature distribution in the chain rule decomposition. As a consequence, it can straightforwardly be applied to problems where the input vector $(X_1, \ldots, X_n)$ contains discrete and continuous variables. It is also convenient for dealing with high dimensional input vectors, while jointly considering the whole set of features in a common distribution is a lot more complex. Finally, training the model is also simplified as each conditional distribution can be optimized separately.

Generative and Discriminative models

If the first goal of the naive Bayes classifier is to assess the class instance of C given a set of observed features, the graph however describes the opposite causal process: for a given class, it seems to generate the corresponding observations. Intuitively, one can understand that instead of encoding knowledge for predicting directly $P(C|X_1, \ldots, X_n)$, the naive Bayes model is built such as describing the process by which the observations are generated. Even if this capability is not always directly exploited, it is common to many popular models which are known as *generative* models. By opposition, a *discriminative* model would only encode the knowledge required to predict $P(C|X_1, \ldots, X_n)$, and is thus unable to generate observations based on a particular instance of its hidden variables.

Intuitively, a generative model provides a valid representation of the real causal processes in the system. For the text categorization application for instance, one can legitimately argue that the word frequencies are, indeed, a direct effect of the actual text category. From a performance perspective however, one may argue that it is unnecessary to embed knowledge about the observation process if our only goal is to estimate the state of the hidden class. Especially, since this knowledge is directly extracted from limited training sets, we should more logically focus on modeling the causal relationship $P(C|X_1, \ldots, X_n)$. In practice, knowing whether it is better to exploit a generative or a discriminative model is a complex problem that is still debated. First of all, the performance of each class of model is deeply correlated to the training method, and the current debates about this issue converged to one main conclusion: generative models usually provide better performance when we learn from small datasets, but they are outperformed by discriminative models when the amount of available training data grows (Ng and Jordan, 2001; Koller and Friedman, 2009). However, generative models are naturally capable of dealing with missing values and incomplete data sets, and some applications may also require to generate observations. Thus, the choice of class of the model is clearly application dependent.

Illustration: The Markov chain

Some applications, like speech recognition, require to model the temporal nature of a system. Sometimes, it is also useful to model any form of sequential data, like the sequence of characters or words in a sentence which does not properly rely on a temporal process. In these domains, the strong correlation between successive events can not be ignored. The most simple and popular model used for expressing such sequentiality is the Markov model.

In the general case, we want each observation y_i of a system to depend on the previous observations. This dependence can be expressed by decomposing the joint distribution $p(y_1, \ldots, y_n)$ in the form

$$p(y_1, \ldots, y_n) = p(y_1) \prod_{i=2}^{n} p(y_i | y_1, \ldots, y_{i-1})$$

Note that this decomposition is derived from successive application of the product rule, and hence always holds for any joint distribution $p(y_1, \ldots, y_n)$.

The prevailing *first-order* Markov model makes the assumption that each observation y_t only depends on the last observation y_{t-1}, and thus that y_{t-1} implicitly encompasses all the information gathered in the past. The graphical model expressing this independence assumption is shown in Fig.2.4.

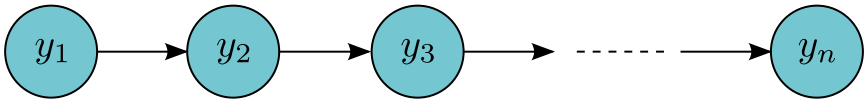


Figure 2.4: First-order Markov chain representing the joint distribution of observations $\{y_1, \ldots, y_n\}$

It is easy to verify that we now have $p(y_i | y_1, \ldots, y_{n-1}) = p(y_i | y_{n-1})$ from the d-separation property applied to evidential trail structures. Thus the d-separation property is the direct graphical equivalent of the first order independence assumption. The joint distribution factorizing over this specific graph can now be written

$$p(y_1, \ldots, y_n) = p(y_1) \prod_{i=2}^{n} p(y_i | y_{i-1})$$

The first-order Markov model is very popular, despite the strong independence assumption that makes the model 'forget' about the past. In some applications, like weather forecasting, it is however important to take into account a sequence of past observations. Generally speaking, many applications may require to assume that the prediction for the next observation

depends more on a trend in the past events than on the single last observation. In that case, one may turn to Markov chains of higher order, allowing the prediction to depend on older observations. An example of second-order Markov chain is depicted in Fig.2.5. Unfortunately, increasing the order of a Markov chain quickly leads to unwieldy models as the number of required parameters for describing the conditional distributions explodes. This is why the Markov chain model is primarily used under the first-order independence assumption.

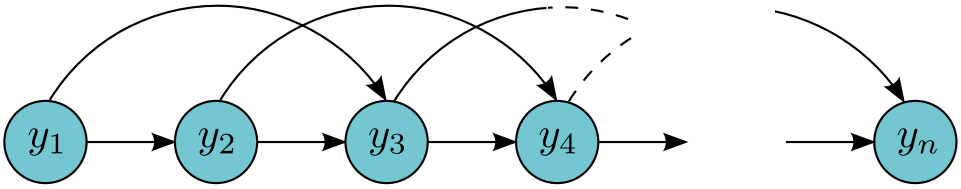


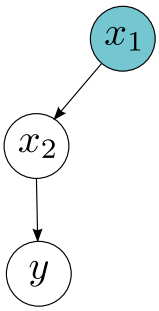
Figure 2.5: Second-order Markov chain graphical model.

2.2 Inference

Once the graphical model and associated conditional distributions are defined, we can subsequently run inference in the graph. The inference task consists in computing the posterior probability distribution of one or multiple variables within the model based on some 'observed' values. Note that in this context the term *observed value* is equally used to refer to an actual measurement (provided by a sensor for instance), but also to refer to a variable that is arbitrarily instantiated to one of its possible values. The inference task can be split in three major operations, or *reasoning patterns*:

Causal reasoning:

Causal reasoning is the most intuitive inference case on a directed graph. It consists in predicting the posterior distribution of a variable, being given an observation in the upstream of the graph. A simple example of this scenario is depicted on the right. In this graph, we want to infer the distribution of y given an observation on x_1, *i.e* $p(y|x_1)$. Being given an observation of x_1, the only distribution we can straightforwardly derive is $p(y, x_2|x_1)$. In order to obtain the conditional distribution of y, one consequently have to marginalize out the variable x_2. In the probabilistic approach, marginalization corresponds to the application of the sum rule to x_2. In a more intuitive interpretation, we simply want to propagate information through x_2, then take into account all its possible values given the observation on x_1. In the continuous case, this is done by integrating the joint distribution over x_2 (sum rule) so that we finally obtain:



$$p(y|x_1) = \int p(y, x_2|x_1)\,\mathrm{d}x_2$$

Evidential reasoning:

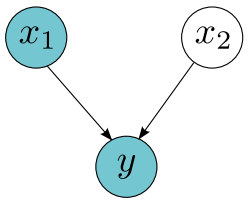
In evidential reasoning, the inference task now consists in propagating information from the effects to the causes. The simplest scenario of evidential reasoning is the one described by the graph depicted on the right. In this graph, a cause x leads to an observed effect y, in a basic generative representation. Thus, given the observation y, we aim at computing $p(x|y)$. However, the graph only describes the conditional distribution $p(y|x)$. Inference then requires to exploit the Bayes' theorem which gives us

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

where $p(x)$ is the prior distribution over x, and $p(y)$ can be computed using the sum and product rule so that $p(y) = \int p(y|x)p(x)\,\mathrm{d}x$. Computing this normalization term is however not always required. Note that the Bayes' theorem gives us the capability to reverse the information flow within the graph.

Intercausal reasoning:

Finally, the last class of reasoning patterns involves information flowing between common causes of a same effect. This is for instance the case in the graph depicted on the right, where both x_1 and x_2 are parents of y. Reminding that this structure is active if y is observed, then being given an observation of x_1 also impacts the posterior distribution of x_2. Intuitively, we know that observing y give us information about x_2. Meanwhile, knowing x_1 also allows us to take away some hypotheses about why the variable y took this specific value, which in turn provides a better intuition about the more probable value of x_1. In this case, inference requires to exploit both marginalization and the Bayes' theorem. Note that intercausal reasoning can be done in any direction, and the required mathematical tools may vary for each case.



Depending on the variables type and on the conditional distribution representation, several methods can be used to perform inference. One may roughly distinguish two main classes of inference methods: *exact inference* methods and *approximate inference* methods. Exact inference can be performed when the variables are discrete, and in the continuous case when the conditional distributions are conveniently chosen to provide closed-form expressions. When the computational complexity of a graph increases, or when no closed-forms solutions exist, one has to turn to approximate inference. Two main methods are worthy of note in this domain. The first method aims at approximating the complex distribution of interest by substituting it with a simpler, tractable distribution. These methods rely on a common optimization process which aims at finding the simple distribution (among a distribution family) that provides the best similarity with the intractable one. The similarity term being provided by the information framework in terms of relative entropy. The second method consists in approximating the distribution over the graph by a set of instantiations to some variables. These methods are usually referred to as particle-based and rely on a simple idea: samples (particles) are generated from parent variables within the graph, starting from the leaf nodes, and are then propagated by simple evaluation of the conditional distributions at these points. An estimation of the resulting posterior distributions is then obtained by combining the particles that were generated for the variables we want to infer.

A very broad range of methods provide solutions to exact and approximate inference, and discussing the details of these approaches is out of the scope of this manuscript. In any cases, the overall inference operation relies

on –and should be remembered as– a simple conceptual principle. That is, inference consists in propagating messages between the parent-child pairs of a graph, so as transmitting the information gathered from the observed variables. Examples of inference method will be provided later in the specific case of the DBNs where exact inference corresponds to the Bayesian filtering recursive equations. The probabilistic regression problem with hidden parameters will also be detailed.

Even if, at first, inference seems to be a distinct and independent tool exploited once the model and its parameters are defined, it is not unusual to perform inference during the training phase. In the presence of hidden variables that remains unobserved, we have to *complete* the missing data via inference before optimizing the parameters of the joint distribution defined over $\mathcal{G}$. Thus, one should not consider inference and learning as two chronologically distinct procedures within the life cycle of a graphical model.

2.3 Learning

In some specific problems, the construction of a PGM can be done after human expertise about the system. For instance, a significant amount of real-life systems exploit a Bayes filter whose conditional distributions are purely handcrafted. However, within the PGM framework, the models can be arbitrarily extended to reach levels of complexity that quickly become intractable for a human expert. In some other applications, we may not even have any useful knowledge about the system. For this reason, and because it is easier and faster to acquire data from the system than to acquire human knowledge about it, the PGM framework was developed with a rich learning machinery. Furthermore, learning the joint distribution from data saves engineering costs, and provides a generic approach for automatically adapting the same model to different applications. Note however that when particular aspects of the real physical system can be described by some deterministic equations, including this knowledge within the model usually increases its performance. Intuitively, this is because combining both system expertise and learning allows to alleviate the learning algorithm and reduce the amount of required modeling power. The training phase then exploits the available training data more efficiently in order to capture the unknown aspects of the system.

As said earlier, some systems are so complex that the structure of the graph $\mathcal{G}$ is left unknown. Then the learning task also consists in extracting an adequate structure from the training data. In this work, models are kept simple enough so that we will never require to explore this area within the learning machinery. More details about structure learning can however be found in (Friedman and Koller, 2003; Koller and Friedman, 2009).

2.3.1 Learning is optimization

Generally speaking, the learning task is viewed as an optimization process whose output is the set of parameters[4] of a joint distribution P factorizing over a graph $\mathcal{G}$. In this perspective, we have to define at first a set of candidate models defining a *hypothesis space*, and a loss function that specifies some constraints and objectives the model should fulfil. For instance, and as will be described hereafter, an intuitive loss function provides a measure of how a model is capable of explaining the observed data, given a specific instance of the distribution parameters.

In this chapter, we mainly discuss two scenarios:

- The data in our training set is fully observed, in which case Bayesian networks usually allow for closed-form solutions.
- The data in our training set is partially observed, and the learning process is generally harder, as we need to take into account the multiple hypotheses about the unobserved variables.

In the next subsections, we discuss the two main methods for learning parameter from data, a task known as *parameter estimation*.

2.3.2 Maximum likelihood parameter estimation

A common principle used for parameter estimation in Bayesian networks is that of *Maximum Likelihood*. To illustrate this principle, let's assume that we are given a training set $\mathcal{D}$ containing M observations of a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ so that $\mathcal{D} = \{\mathcal{X}[1], \ldots, \mathcal{X}[M]\}$. We also assume that we defined a PGM $\{\mathcal{G}, P_{\mathcal{G}}(\mathcal{X} : \Theta)\}$ with Θ the set of parameters of the distribution P. Note that we are in the case of fully observed data.

Thus, the probability distribution of each instance $\mathcal{X}[m]$ within the dataset is given by $P_{\mathcal{G}}(\mathcal{X}[m] : \Theta)$. We can view this last term as a function of the parameter vector Θ assessing how likely it is for the distribution $P_{\mathcal{G}}$ to explain the observed data $\mathcal{X}[m]$. This term is then referred to as the likelihood function for $\mathcal{X}[m]$.

If we consider that the variables instances $\{\mathcal{X}[1], \ldots, \mathcal{X}[M]\}$ are independent and were sampled from a common underlying distribution that we try to model through $P_{\mathcal{G}}$, these instances are called *independent and identically distributed* (i.i.d). Thanks to this independence assumption we can derive a simple expression for the likelihood function $\mathcal{L}$ of the dataset $\mathcal{D}$ through the product

$$L(\Theta : \mathcal{D}) = \prod_{m=1}^{M} P_{\mathcal{G}}(\mathcal{X}[m] : \Theta) \tag{2.4}$$

[4]Note that in the case of structure learning the output also contains a representation of the graph structure.

Now taking into account the specific chain rule decomposition for $P_{\mathcal{G}}(\mathcal{X} : \Theta)$ over $\mathcal{G}$, we can derive a final expression for the likelihood function that applies to any Bayesian network:

$$\begin{aligned} L(\Theta : \mathcal{D}) &= \prod_{m=1}^{M} P_{\mathcal{G}}\big(\mathcal{X}[m] : \Theta\big) \\ &= \prod_{m=1}^{M} \prod_{i=1}^{n} P\big(X_i[m] | Pa^{\mathcal{G}}_{X_i[m]} : \Theta\big) \\ &= \prod_{i=1}^{n} \Big(\prod_{m=1}^{M} P\big(X_i[m] | Pa^{\mathcal{G}}_{X_i[m]} : \Theta\big) \Big) \end{aligned} \tag{2.5}$$

Once the likelihood function defined, the set of parameters explaining at best the training data $\mathcal{D}$ can thus be found by choosing the parameters Θ^* that verify

$$L(\Theta^* : \mathcal{D}) = \max_{\Theta} L(\Theta : \mathcal{D}) \tag{2.6}$$

This means that we have to ensure that $P_{\mathcal{G}}(\mathcal{X}[m] : \Theta)$ is continuous and differentiable with respect to Θ. Besides, it is customary to maximize the log of the likelihood function, then called *log-likelihood*, as it often simplifies the mathematical expression for $\mathcal{L}$. Since the likelihood forms a product of a large amount of probabilities, each of which being potentially small, the log likelihood also helps in providing a better numerical precision and stability. Note that the log function being monotonically increasing, maximizing the log-likelihood is equivalent as maximizing the likelihood function.

Interestingly, we see that the likelihood of a Bayesian network forms a product of local likelihood for each node X_i taking the form:

$$\prod_{m=1}^{M} P\big(X_i[m] | Pa^{\mathcal{G}}_{X_i[m]} : \Theta\big)$$

If the parameters involved in the computation of each local likelihood form disjoints subsets Θ_i, then the likelihood function can be seen as a product of local terms which can be independently maximized. Thus maximizing the log of the likelihood function (2.5) consists in subsequent optimizations of

$$\ell(\Theta : \mathcal{D}) = log(L(\Theta : \mathcal{D})) = \sum_{i=1}^{n} \Big(\prod_{m=1}^{M} P\big(X_i[m] | Pa^{\mathcal{G}}_{X_i[m]} : \Theta_i\big) \Big) \tag{2.7}$$

with respect to Θ_i. This decomposition usually leads to simple solutions for the optimization problem if the conditional distributions are conveniently chosen.

2.3.3 Bayesian parameters estimation

The likelihood function defines an intuitive and powerful loss function for optimizing the model parameters. However, it is well known that its straightforward optimization suffers from the typical drawbacks of the *frequentist* interpretation of probabilities. Practically, it means that the parameter optimization blindly relies on the sole training data and ignores any prior knowledge we may have about the system. Also, it does not include uncertainty over the parameters learned from data in its basic formulation. Consequently, the learned models are prone to overfit the training set and to perform poorly on new data. These problems are however automatically tackled by revisiting the likelihood optimization problem in a full Bayesian treatment.

Following the Bayesian approach, the model parameter vector Θ is now represented by an additional random variable, so that the likelihood is now given by the distribution $P(\mathcal{D}|\Theta)$. An estimation of the parameters can now be obtained by straightforward exploitation of the Bayes rule:

$$P(\Theta|\mathcal{D}) = \frac{P(\mathcal{D}|\Theta)P(\Theta)}{P(D)} \tag{2.8}$$

where $P(\Theta)$ is the prior distribution over the parameters and $P(D)$ can be derived by marginalizing out Θ in the likelihood function:

$$P(\mathcal{D}) = \int P(\mathcal{D}|\Theta)P(\Theta)\,\mathrm{d}\Theta$$

Unlike the maximum likelihood approach which provides a point estimate Θ^*, the Bayesian treatment defines Θ in terms of probabilities throughout the optimization process. Before optimization, the initial belief and uncertainty about Θ is defined by the prior distribution $P(\Theta)$. This knowledge is jointly taken into account along with the likelihood of the training set in order to compute a posterior belief about the parameters. In a typical Bayesian approach, this posterior distribution is proportional to the product of the likelihood and the prior, as described in (2.8).

In some cases, the prior and the conditional distributions within the model are chosen so that we can find a closed-form expression for the posterior. Ensuring that the prior and the likelihood are conjugate distributions is a common practice for obtaining such convenient posteriors. Note that in this case, the prior and the posterior distributions have the same functional form, so that a previously computed posterior can be used straightforwardly as the prior for a new training phase. When there exists no conjugate prior, or when alternative priors are preferred for their specific characteristics, one can approximate the fully Bayesian approach with the *Maximum a posteriori* (MAP) point estimate. Namely, we seek for the parameters that maximize

the posterior distribution:

$$\Theta^{MAP} = \underset{\Theta}{\text{argmax}} \left(log\, P(\Theta|\mathcal{D}) \right) \tag{2.9}$$

To see how MAP estimation differs from the maximum likelihood method, we can expand (2.9) using (2.8) to get:

$$\begin{aligned} \Theta^{MAP} &= \underset{\Theta}{\text{argmax}} \left(log\, P(\Theta|\mathcal{D}) \right) \\ &= \underset{\Theta}{\text{argmax}} \left(log\, \frac{P(\mathcal{D}|\Theta)P(\Theta)}{P(\mathcal{D})} \right) \\ &= \underset{\Theta}{\text{argmax}} \left(logP(\mathcal{D}|\Theta) + logP(\Theta) \right) \end{aligned}$$

Thus, the MAP estimate still tries to maximize the (log)likelihood function, now biased by the prior distribution over Θ. This last term is known for fostering regularization, *i.e* it prevents the optimization process to converge to extreme values that would only fit the training set. This behaviour is known as *overfitting*, and subsequently leads the model to perform poorly on new data. In the worst case, the possible instances of the random variables are only partly observed in the training set. This leaves the purely frequentist approach unable to draw any conclusion about these specific cases, and consequently to provide reliable parameter estimates. On the other hand, the Bayesian approach rather relies on the prior knowledge introduced through $P(\Theta)$ when the training set provides insufficient information. Recalling from (2.7) that the log-likelihood grows linearly with the number of observations in $\mathcal{D}$, the prior influence however remains inversely proportional to the size of the training set. As such, the Bayesian estimation method usually provides parameters that generalize better, but care has to be taken when defining the prior distributions.

The overfitting issue:
Overfitting remains one of the major issue when learning a Bayesian network. Besides the case of small training sets, and potentially non identically distributed observations, the model complexity is also to be considered as a main cause for unsuccessful parameter estimation. In practice, when we increase the number of parents $Pa^{\mathcal{G}}_{X_i}$ of a variable in the graph, we automatically increase the amount of required training data. This is because the number of possible instances of the parent set increases exponentially with the number of parents. Consequently, even large training sets may not provide a sufficient number of samples for each possible instance. This lack of information, relatively to the number of possible instances for the variables $\mathcal{X}$ in the graph, usually leads to overfitting. Thus, as can be expected, increasing the modeling power of a model is not always a key to better performance, as it may require too large amounts of data to be properly trained.

The strength of the Bayesian estimation method however lies in the representation of Θ through a complete probabilistic distribution. When a proper closed-form expression for the posterior can be derived, we can use this distribution instead of a point estimate in the subsequent inference tasks. This representation conveys a more realistic representation of the parameters including our uncertainty in the estimate. To illustrate this fundamental difference, we can imagine that a parameter is now represented by a function which get more sharply peaked around the optimal value when the amount of data in the training set[5] increases. On the negative side, exploitation of the model is consequently more complex as we have to integrate out all the parameters in any manipulation involving Θ.

To illustrate this principle, we assume a simple graph containing a parent variable X and a child Y. The joint distribution $P(X, Y|\Theta)$ then decomposes in the product:

$$P(X, Y|\Theta) = P(X|\Theta_X)P(Y|X, \Theta_Y)$$

with $\Theta = \{\Theta_X, \Theta_Y\}$, where Θ_X is the set of parameters of the prior distribution over the parent variable X, and Θ_Y the parameters of the conditional distribution over Y.

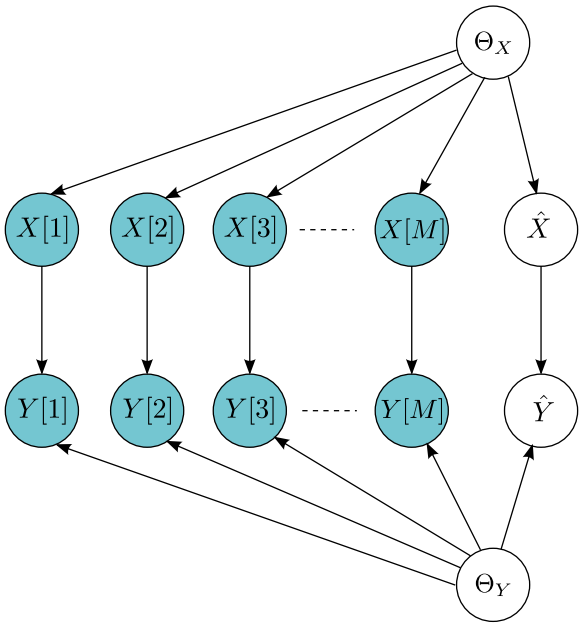


Figure 2.6: Ground Bayesian Network of a two-variable model: each instance $\{X[i], Y[i]\}$ within the training set is depicted as an independent sample, along with the predicted samples $\{\hat{X}, \hat{Y}\}$. Note that the parameters $\Theta = \{\Theta_X, \Theta_Y\}$ are shared by all instances of X and Y.

[5]And confirming this trend around a single optimal value

If our task now consists in predicting new instances $\hat{X}$ and $\hat{Y}$ given the training data $\mathcal{D}$ and a posterior distribution $P(\Theta|\mathcal{D})$, we can write

$$\begin{aligned} P(\hat{X}, \hat{Y}|\mathcal{D}) &= \int P(\hat{X}, \hat{Y}|\mathcal{D}, \Theta) P(\Theta|\mathcal{D}) \, \mathrm{d}\Theta \\ &= \int P(\hat{X}, \hat{Y}|\Theta) P(\Theta|\mathcal{D}) \, \mathrm{d}\Theta \end{aligned}$$

where we use d-separation to obtain the last term, as all instances $\{X[i], Y[i]\}$ are independent given the common cause variable Θ. This independence assumption is illustrated in the ground Bayesian network depicted in Fig.2.6.

So far, we assumed that the prior distribution over Θ was provided. Practically, we need to define a distribution over a continuous space that represents our available knowledge about the parameters before any observation on the real system. If we do not have any knowledge about the parameters, we can use *uninformative* priors, and then represent Θ with a uniform distribution in a given interval. If we want to be more specific about the shape of the prior distribution, we usually introduce a family of priors represented by a parametric function $P(\Theta : \alpha)$ where the variables $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ are called the *hyperparameters* of the model. As said previously, the prior distribution is also chosen for its functional form, so that the posterior $P(\Theta|\mathcal{D})$ forms a compact and convenient analytic expression. When such parametric priors are used, the learning task then consists in finding the optimal values for $\{\alpha_1, \ldots, \alpha_n\}$.

In practice, a fully Bayesian treatment of the parameter estimation problem requires to define *hyperprior* distributions over the hyperparameters. The marginalization of both Θ and α that would necessarily ensue is however often intractable. Various methods exist to get around this problem via approximation. A common approach follows a similar idea as in the MAP estimation, and then approximates α with point estimate. This method is referred to as *Type-2 maximum likelihood* or *evidence approximation* and will be described in chapter 5. Alternatively, one may proceed to exact marginalization over one of the two parameters, and then approximate the resulting posterior that can subsequently be integrated in closed-form. This can be done with more advanced techniques such as Laplace approximation (Chickering and Heckerman, 1997), or variational methods (MacKay, 1997). Finally, one can view the parameters as new hidden variables that are clearly never observed. This specific problem is then addressed with dedicated learning methods that are built to deal with incomplete data. This specific task is discussed in the next subsection.

2.3.4 Learning with incomplete data

The two former approaches for learning the parameter assumed that the set of variables depicted by the model was fully observed in the training set, *i.e*

we obtained observations for each variable X_i in $\mathcal{X}$. The ability to observe every variable is however quite unlikely in real applications, and as depicted in the previous subsection, the fully Bayesian treatment of the parameter estimation unavoidably introduces new hidden variables representing the parameters. Many problems arise from the missing data. To be more specific, let us denote the set of i.i.d observed variables by $O[m]$ and the missing variables by $H[m]$. The likelihood function is now provided by:

$$\begin{aligned} L(\Theta : \mathcal{D}) &= \prod_{m=1}^{M} P(O[m]|\Theta) \\ &= \prod_{m=1}^{M} \int P(O[m], H[m]|\Theta) \, \mathrm{d}H[m] \end{aligned}$$

This expression shows us that the learning task now requires to run inference on the missing (hidden) variables. Usually, the resulting likelihood does not admit a closed-form expression, and the decomposition (2.7) allowing for easy optimization of the parameters, is not suitable.

In a more intuitive understanding of the issues caused by incomplete data, one may view the training data as a set of constraints in the parameter optimization task. Consequently, an incomplete training set yields a lack of constraints in the optimization problem, which in turn decreases the capability of finding a unique set of parameters. Multiple methods provide a solution to this problem. One may for example turn to an adapted gradient ascent method or alternatively perform sampling to approximate the parameter values (Koller and Friedman, 2009). We shall however focus on a specific approach known as *Expectation Maximization* that we will exploit later.

Expectation Maximization

When direct optimization of the complete data likelihood $P(O[m], H[m]|\Theta)$ is substantially easier than optimization of the partially observed data likelihood, a simple approach consists in artificially assigning the missing variables with adequate values. We know that for a given instance of Θ, we can easily run inference in the model to evaluate the missing variables. This task is however impossible as we are concurrently trying to learn the model parameters. The Expectation Maximization algorithm (EM) (Dempster et al., 1977) provides an elegant solution to this problem by repeating two steps: in the first step, we use the current value of the parameters to infer the missing variables. In a second step, we learn a new set of parameters based on these artificial observations and the incomplete training set. Repetition of these two steps provides interesting convergence guarantees, as it can be formally proved that every time we get a new parameter Θ that differs from the previous one, the associated likelihood $P(O|\Theta)$ increases.

As seen in the algorithm summary below, the first step does not only require to infer the posterior distribution $P(H|O,\Theta^{old})$. In practice, the purpose of the fist step is to compute the expectation of the complete log-likelihood with respect to the current posterior $P(H|O,\Theta^{old})$ and for any parameter Θ. For this reason this step is referred to as the *E step* (Expectation). In the subsequent *M step* (Maximization) we then find the optimal parameter maximizing the expected likelihood. Note that the introduction of the expectation yields the convergence properties of the EM algorithm. A formal proof of this property can be found in (Bishop, 2006).

Expectation Maximization algorithm:

Let $\mathcal{D} = O$ be the set of observed variables, H the set of hidden variables, and Θ the set of all model parameters.

1. Set initial value for parameters Θ^{old}
   Set a convergence threshold ϵ.

2. **E step** (Expectation)

   Evaluate $P(H|O,\Theta^{old})$ and compute

$$\mathcal{Q}(\Theta,\Theta^{old}) = \int ln\big(P(H,O|\Theta)\big)\, P(H|O,\Theta^{old})\,\mathrm{d}H$$

3. **M step** (Maximization)

$$\Theta^{new} = \underset{\Theta}{\operatorname{argmax}}\, \mathcal{Q}(\Theta,\Theta^{old})$$

4. If $|\Theta^{new} - \Theta^{old}| > \epsilon$,
   Set $\Theta^{old} \leftarrow \Theta^{new}$ and go to step 2.

In the general formulation of the EM algorithm, the E step performs what can be seen as *soft* completion of the data, by integrating over the posterior of the missing variables. An alternative solution consists in evaluating a point estimate $H[m]$ of the hidden variables and subsequently to optimize the likelihood of the complete data $\{H,O\}$. The first step then approximates the expectation of the complete data likelihood with *hard assignments* of the missing variables. This alternative approach is thus referred to as *hard-assignment EM*, and may provide different results than the general EM algorithm. However, it remains very useful in some cases when the computation of $\mathcal{Q}(\Theta,\Theta^{old})$ is too complex. Usually, the hard-assignment

EM algorithm converges faster as it progresses through discrete steps. It is also known to be prone to oscillations in the last steps of the optimization, and thus requires to monitor the evolution of the likelihood over the training set after each iteration of the EM steps.

2.3.5 Generative and discriminative training

The purpose of a Bayesian network is to approximate the real joint distribution[6] over the system variables $\mathcal{X}$. This gives us the ability to generate any variable $X_i \in \mathcal{X}$. In many applications however, we aim at predicting a subset of variables X given observations Y where $\mathcal{X} = \{X, Y\}$. In this case, we are only interested in modeling the distribution $P(X|Y)$. As seen earlier, this can be done by training a discriminative model whose only purpose is to approximate this conditional distribution. Such a goal-oriented training is called *discriminative training*. Note that discriminative models do not allow us to generate any variable in $\mathcal{X}$.

Alternatively, we can also use a generative model to evaluate the prediction distribution $P(X|Y)$, as it is the case for Naive Bayes classifiers. Generative models being conceptually designed to represent a joint distribution, they are usually trained through *generative training*, *i.e* such as modeling $P(X, Y)$. However, a generative model can be optimized with respect to any relevant loss function. Thus we can optimize the parameters Θ of a generative model so that they optimize its performance at predicting $P(X|Y)$, which corresponds to to discriminative training of a generative model.

As said before, this approach is unusual as it changes the meaning of the parameters (Koller and Friedman, 2009), and consequently the interpretation of the model. For instance, each conditional distribution $P(X_i|Pa_{X_i})$ taken independently is not tuned such as explaining at best the observations made about the effect X_i given a cause Pa_{X_i}. The conditional distribution is now optimized so as we get the best performance at predicting an arbitrary subset of variables within the graph. Also, the simple learning methods discussed earlier do not apply in this case, potentially leading to more challenging computations. However, as long as we do not use the model for a different task than the one we discriminatively trained it for, this approach proved to help in compensating for the various assumptions made by describing a generative model. Note that a discriminative model usually makes fewer assumptions over the system, as it straightforwardly aims at describing a subaspect, and not an exhaustive set of causal interactions.

[6] Supposing this distribution actually exists.

2.4 Conclusion

The PGM framework is a very powerful tool for formulating, learning, and exploiting complex models through the Bayesian approach. As a very generic tool, it allows to re-interpret many problems through a probabilistic analysis and to intuitively enhance existing models thanks to the handy diagrammatic representation. For instance, we will see in the next chapters that we can easily derive new dynamic networks by augmenting the standard state-observation model. This implicitly restricts our analysis to a specific class of graphical models known as the Bayesian networks, in which causal relationships are directed and indicated by arrows. The other class of graphical models, namely the Markov Random fields in which the links are undirected, are not considered in this work.

The PGM framework is particularly useful in that insights into the system can easily be understood through a quick analysis of the graph. This representation notably allows for a sound analysis of the dependencies between variables, which would appear more intricate without this visual representation.

An interesting aspect we try to point out in this work concerns the connexion between the dependencies encoded through the graph, and the training method we chose. In other words, the objective function that is optimized for learning the model parameters may drastically change the meaning of the model. This peculiar aspect is usually ignored in most of the implementations, where the system representation and its training are usually considered as two distinct problems. In the next chapter we will see how the specific discriminative training changes the dependencies –and more importantly the independences– expressed in the model, and how we can take advantage of it.

Chapter 3

The state-observation model: background and extensions

Contents

This chapter intends to provide a deeper understanding of inference and learning in the state-observation model. At first, we address inference and detail the well-established recursive filtering method providing a simple and fast solution for evaluating the posterior state belief given the past observations. We subsequently give a background of traditional and more recent methods for learning the model. Based on this knowledge and on the notorious issues when learning the model with the popular approaches, two central contributions of this thesis are presented. At first, we describe a new model for the observation distribution that incorporates an explicit representation of measurement alterations. Second, we suggest the exploitation of an alternative *discriminative* loss function when learning the parameters. A new interpretation of this training method is presented, based on the duality between discriminative training and discriminative modeling. As such, these contributions are to be seen as elemental modeling principles, which will lead to the implementations depicted in the next chapters.

3.1 DBNs and the state-observation model

The first extension of the PGMs for modeling the temporal nature of a system was introduced in (Dean and Kanazawa, 1989). The term *Dynamic Bayesian Networks* (DBNs) was introduced later by the same authors. Originally, the DBNs are built upon the Markov and time-homogeneous assumption previously mentioned. In this section, we do not provide a general definition of the DBNs, but rather focus on a specific instance of DBN that is underlying any Bayes filter implementation: the state-observation model. A more formal definition of the DBNs and a description of their different classes can be found in (Koller and Friedman, 2009).

Based on the first-order Markov assumption, a state-observation model represents the evolution in time of a set of hidden state variables x_t through a Markov chain with stationary distribution $P(x_t|x_{t-1})$. For each hidden state, an observation is made and modeled by an observation distribution $P(y_t|x_t)$. Three distinct inference tasks are usually performed over a state-observation model, namely filtering, prediction and smoothing:

- Filtering consists in estimating the distribution of the current state x_t given all past observations up until time t.
- In the prediction task, we want to evaluate the distribution of a future state x_{t+n} given observations $\{y_1, \ldots, y_t\}$.
- Finally, smoothing consists in computing the distribution of a state x_k given a set of observations $\{y_1, \ldots, y_T\}$ with $k < T$.

These different tasks are illustrated in Fig.3.1.

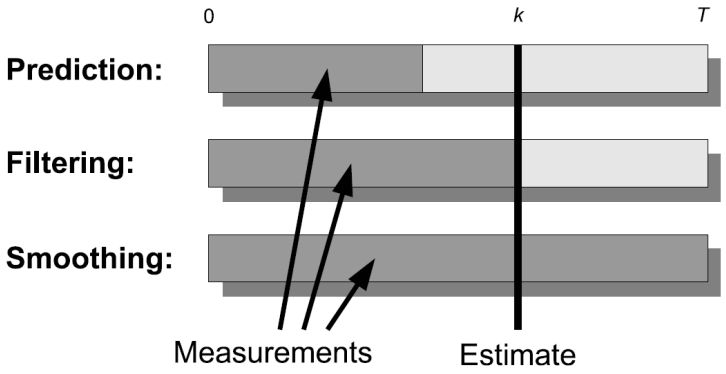


Figure 3.1: The three common inference problems in the state-observation model: prediction, filtering and smoothing. (Excerpt from (Särkkä, 2013, p.11))

Note that in some specific applications, like speech recognition for instance, it is more important to find the most probable *sequence* given a set of observations. This problem strongly differs from the three former tasks, which provide the most probable *state* at each time step, taken individually. Generally, the latter problem of finding the most probable sequence is solved with the Viterbi algorithm (Viterbi, 1967).

3.2 Inference in the state-observation model

In this subsection, we focus on the filtering task which is our main concern. In a purely probabilistic treatment, inference equations are derived for the LDS model, in which both the prediction and observation distributions are linear Gaussian. During these developments, we will show how inference leads to the basic Bayesian filtering recursive equations. As expected, further development of this equation in the specific case of the LDS model leads to the well known Kalman filter algorithm. Subsequently, we discuss the non-Gaussian and non-linear cases, and detail an efficient algorithm for non-linear models where the assumption of Gaussian noise is still reasonable. Note that the theoretical developments made in this thesis require the state belief to be represented by a Gaussian distribution, whether it is exact or approximate, and the specific case of non-Gaussian distributions is left as a matter for future work.

3.2.1 Recursion equations

Transition and observation distributions.

The LDS model relies on three simple equations:

$$\begin{aligned} p(x_t|x_{t-1}) &= \mathcal{N}(x_t|Ax_{t-1}, \Sigma_d) && \text{(transition)} \\ p(y_t|x_t) &= \mathcal{N}(y_t|Cx_t, \Sigma_o) && \text{(observation)} \\ p(x_1) &= \mathcal{N}(x_1|\mu_1, \Sigma_1) && \text{(initial state and uncertainty)} \end{aligned}$$

where x_t is a vector of hidden variables of size n, y_t a vector of observations of size m, A an $n \times n$ matrix representing the linear transition model, C an $m \times n$ matrix defining the linear observation model, Σ_d an $n \times n$ matrix defining the Gaussian noise over the system dynamics, and Σ_o an $m \times m$ matrix defining the Gaussian noise associated with the observation process. Also, the initial state x_1 is described by a Gaussian distribution centred around μ_1 with initial uncertainty Σ_1. As described in Fig.3.2, the state observation model represents the joint distribution over the latent state variables and associated observations $\{x_t, y_t\}_{t=1}^{T}$, where the horizontal edges depict the transition distribution, and the vertical edges the observation distribution.

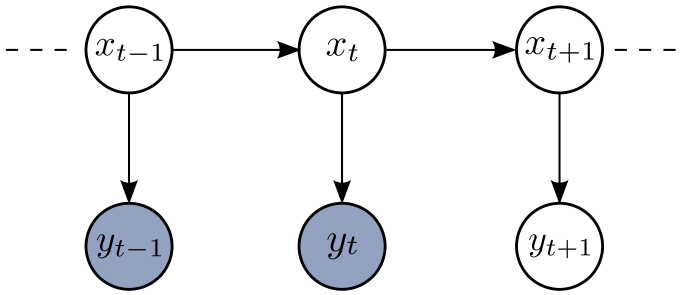


Figure 3.2: State-observation model.

Recursion equations.

We first define the quantity $\alpha(x_t)$ representing the joint probability of x_t and all the observations made up to time t:

$$\alpha(x_t) \equiv p(x_t, y_1, \ldots, y_t)$$

In a first step, the observation component can be simply extracted from $\alpha(x_t)$ as:

$$\begin{aligned} \alpha(x_t) &= p(x_t, y_1, ..., y_t) \\ &= p(y_1, ..., y_t|x_t)p(x_t) \\ &= p(y_1, ..., y_{t-1}|y_t, x_t)p(y_t|x_t)p(x_t) \end{aligned}$$

Using d-separation on the graphical model we can prove that the joint observations made up to time $t-1$ are independent of the observation at time t given the hidden variable x_t. Thus $p(y_1, ..., y_{t-1}|y_t, x_t) = p(y_1, ..., y_{t-1}|x_t)$.

Based on this independence property, a recursive relation can be derived:

$$\begin{aligned}
\alpha(x_t) &= p(y_1, ..., y_{t-1}|x_t)p(y_t|x_t)p(x_t) \\
&= p(y_t|x_t)p(y_1, ..., y_{t-1}, x_t) \\
&= p(y_t|x_t) \int p(y_1, ..., y_{t-1}, x_{t-1}, x_t)\, \mathrm{d}x_{t-1} \\
&= p(y_t|x_t) \int p(y_1, ..., y_{t-1}, x_t|x_{t-1})p(x_{t-1})\, \mathrm{d}x_{t-1} \\
&= p(y_t|x_t) \int p(y_1, ..., y_{t-1}|x_t, x_{t-1})p(x_t|x_{t-1})p(x_{t-1})\, \mathrm{d}x_{t-1}
\end{aligned}$$

Using d-separation again we can easily understand that $p(y_1, ..., y_{t-1}|x_t, x_{t-1}) = p(y_1, ..., y_{t-1}|x_{t-1})$ as there is no active trail between y_{t-1} and x_t given x_{t-1}.

$$\begin{aligned}
\alpha(x_t) &= p(y_t|x_t) \int p(y_1, ..., y_{t-1}|x_{t-1})p(x_t|x_{t-1})p(x_{t-1})\, \mathrm{d}x_{t-1} \\
&= p(y_t|x_t) \int p(y_1, ..., y_{t-1}, x_{t-1})p(x_t|x_{t-1})\, \mathrm{d}x_{t-1} \\
&= p(y_t|x_t) \int \alpha(x_{t-1})p(x_t|x_{t-1})\, \mathrm{d}x_{t-1}
\end{aligned}$$

In practice the conditional distribution $p(x_t|y_1, ..., y_t)$ provides a better numerical stability than the joint distribution $p(x_t, y_1, ..., y_t)$ since it decreases the dimensionality of the represented space. Also recalling that we aim at evaluating $p(x_t|y_1, ..., y_t)$ at each time step, the recursive relation can be easily adapted by normalizing $\alpha(x_t)$ so that we now work with

$$\hat{\alpha}(x_t) = p(x_t|y_1, ..., y_t) = \frac{\alpha(x_t)}{p(y_1, ..., y_t)}$$

Also introducing a scaling factor $c_t = p(y_t|y_1, ..., y_{t-1})$, we finally have:

$$\begin{aligned}
c_t\hat{\alpha}(x_t) &= p(y_t|y_1, ..., y_{t-1})\frac{\alpha(x_t)}{p(y_1, ..., y_t)} \\
&= \frac{\alpha(x_t)}{p(y_1, ..., y_{t-1})}
\end{aligned}$$

Thus

$$c_t \hat{\alpha}(x_t) = p(y_t|x_t) \int \hat{\alpha}(x_{t-1}) p(x_t|x_{t-1}) \, \mathrm{d}x_{t-1} \tag{3.1}$$

Note that (3.1) holds for any state-observation model without loss of generality since no particular properties of the observation and prediction distribution have been exploited in the previous derivation. As such, (3.1) defines the generic recursive relationship used in any Bayes filter.

Intuitively, this equation shows that the recursive estimation of the belief over the state x_t relies on two fundamental steps. The first step involves the computation of a *prediction* term that corresponds to the propagation of the prior belief x_{t-1} through the dynamic model (the integral over x_{t-1}). Subsequently, the new belief over x_t is multiplied by the probability that the observation y_t is made at the position depicted by the hypothetical predicted state belief. This product is usually referred to as the *measurement update*. To summarize, the Bayes filter starts by 'blindly' propagating the previous state belief through the dynamic model, and then corrects this prediction given the observation y_t. We now turn to the specific case of the LDS model and show how the posterior belief over x_t can be obtained through elegant closed-form expressions.

3.2.2 Inference in LDS: The Kalman filter equations

Based on the conjugate properties of linear Gaussian models provided in appendix A, and since the distribution over the initial state $p(x_1)$ is Gaussian, we know that the integral over x_{t-1} in the right term of (3.1) leads to a new Gaussian distribution. Jointly with the linear Gaussian observation distribution $p(y_t|x_t)$, this in turns gives a posterior distribution over x_t that is also Gaussian. This property ensures that there is a closed form solution for the recursive equation, and that all the distributions $\hat{\alpha}(x_t)$ will be Gaussian. As a consequence, we denote :

$$\begin{aligned} \hat{\alpha}(x_t) &= p(x_t|y_1, ..., y_t) \\ &= \mathcal{N}(x_t|\mu_t, V_t) \end{aligned}$$

And the recursive equation (3.1) now takes the form:

$$c_t \mathcal{N}(x_t|\mu_t, V_t) = \mathcal{N}(y_t|Cx_t, \Sigma_o) \int \mathcal{N}(x_{t-1}|\mu_{t-1}, V_{t-1}) \mathcal{N}(x_t|Ax_{t-1}, \Sigma_d) \, \mathrm{d}x_{t-1} \tag{3.2}$$

Noticing that the integral in (3.2) corresponds to the marginalization of x_{t-1} in the prediction distribution, we exploit the linear Gaussian property (A.1) discussed in appendix A to write:

$$\int \mathcal{N}(x_{t-1}|\mu_{t-1}, V_{t-1})\mathcal{N}(x_t|Ax_{t-1}, \Sigma_d)\,\mathrm{d}x_{t-1} = \mathcal{N}(x_t|A\mu_{t-1}, P_{t-1}) \quad (3.3)$$

where $P_{t-1} = \Sigma_d + AV_{t-1}A^t$.

Because $\mathcal{N}(x_t|A\mu_{t-1}, P_{t-1})$ can be seen as our prior over x_t, we can make use of (A.1) and (A.2) to evaluate c_t and the inverse conditional distribution $p(x_t|y_t)$ respectively, where both distributions are implicitly conditioned on $\{x_1, \ldots, x_{t-1}\}$. Note that we do not properly exploit the equality (3.2), but rather proceed to an operation called *completing the square*. In practice, we know that the right side of (3.2) is a product of exponential functions where part of the exponent terms correspond to the mean and variance of the distribution over x_t. Completing the square then consist in identifying the proper components μ_t and V_t, provided in closed-form by (A.2). Thus we have:

$$p(x_t|y_t, x_1, \ldots, x_{t-1}) = \hat{\alpha}(x_t) = \mathcal{N}(x_t|\mu_t, V_t) \quad (3.4)$$

with

$$\begin{aligned}\mu_t &= (P_{t-1}^{-1} + C^t\Sigma_o^{-1}C)^{-1}(C^t\Sigma_o^{-1}y_t + P_{t-1}^{-1}A\mu_{t-1}) \\ V_t &= (P_{t-1}^{-1} + C^t\Sigma_o^{-1}C)^{-1}\end{aligned}$$

Using the identity (A.3) we can write

$$(P_{t-1}^{-1} + C^t\Sigma_o^{-1}C)^{-1}C^t\Sigma_o^{-1}y_t = P_{t-1}C^t(CP_{t-1}C^t + \Sigma_o)^{-1}y_t$$

and from the identity (A.4)

$$\begin{aligned}&(P_{t-1}^{-1} + C^t\Sigma_o^{-1}C)^{-1}P_{t-1}^{-1}A\mu_{t-1} \\ &= [P_{t-1} - P_{t-1}C^t(\Sigma_o + CP_{t-1}C^t)^{-1}CP_{t-1}]P_{t-1}^{-1}A\mu_{t-1} \\ &= A\mu_{t-1} - P_{t-1}C^t(\Sigma_o + CP_{t-1}C^t)^{-1}CA\mu_{t-1}\end{aligned}$$

Thus we finally have:

$$\begin{aligned}&\mu_t = A\mu_{t-1} + K_t(y_t - CA\mu_{t-1}) \\ &V_t = (I - K_tC)P_{t-1}\end{aligned}$$

Where we defined the *Kalman gain matrix*:

$$K_t = P_{t-1}C^t(\Sigma_o + CP_{t-1}C^t)^{-1}$$

These equations define the Kalman filtering algorithm, providing a simple and fast solution for evaluating the normalized distribution $p(x_t|y_1, ..., y_t)$

at any time step t. Fundamentally, the succinctness of the state representation is due to the conjugate property of the linear Gaussian system that allows for closed form expressions for all the intermediate distributions at any time step. That is, the prior over the state obtained after propagating the last estimate through the prediction model, the observation likelihood, and the new estimate are all Gaussian, and thus simply defined by their mean and variance.

3.2.3 Inference in non-linear and non-Gaussian models

In the last section, the derivation of exact and simple analytical expressions for the filtering inference task was possible thanks to the linear Gaussian assumption made about the state and the observation variables. However, it is very common to rely on non-linear model for both prediction and observation, as many real systems rely on complex functions for the dynamics and the mapping from the state to the observation. Note that the case of multi-modal distributions exploited for instance in multi-target tracking will not be discussed here and in further chapters.

Similarly, non-Gaussian noise models might also be preferred, as for example the measurement generation process may clearly not present a normally distributed uncertainty. Many of the complex noise models exploited in the observation process are notably meant to better represent the occurrence of outliers. For example, and as suggested in (Loxam and Drummond, 2008), the *Student-t* distribution is an interesting alternative to the Gaussian model that provides better robustness regarding the presence of outliers. From a Bayesian point of view, the Student-t distribution is obtained by introducing a conjugate prior distribution for the precision (or inverse of the covariance) of a Gaussian. After marginalization this results in a Gaussian-like distribution with *heavy tails*. Consequently the distribution inherently allows for the presence of a limited number of outliers samples in the training set. Put into our research context, one should argue that the exploitation of such models should be preferred to the prevailing Gaussian distribution. We however prefer to tackle the presence of outliers through a dedicated additional model, and thus relax the observation noise model whose role only consists in exploiting at best a pre-selected subset of observations. The exploitation of a proper conjugate prior for the observation distribution is still to be considered for future work, the purpose of this study being to prove at first the applicability of a context-dependent observation model.

In addition, non-Gaussianity does not only arise from non-Gaussian noise models. In fact, when the prediction and observation models are non-linear, an initial Gaussian state prior does not ensure that the further state distributions will be Gaussian in turn. Consequently, and in any of these cases, one has to turn to approximations methods since most of the time such models do not lead to tractable inference.

Approximate inference methods

We can roughly identify two distinct families of approximation methods, depending on the modeling assumptions that are made. The first family addresses the case of non-linear prediction and observation functions while the assumption of Gaussian noise is still reasonable. It is then possible to exploit some *deterministic approximations* which yields the well known *Extended Kalman filter* and the *Unscented Kalman filter*. When non-linear and non-Gaussian models are required, a popular family of approximate inference methods is the particle-based approach, referred to as *particle filtering* or *sequential Monte-Carlo* approaches. These approximate methods are more precisely described in Appendix B.

In the following theoretical developments, we prefer to focus at first on problems which for the state belief can be directly represented by an analytic expression, and not by a set of particles. Exploitation of this latter representation requires a re-examination of the proposed approaches that is kept for future work. This includes the study on the conditions under which a set of particles converges to the true distribution. Thus, we only consider linear and non-linear models that allow for deterministic approximation. Consequently we will assume that the state distribution can reasonably be represented by a Gaussian. Note that this assumption greatly simplifies derivation of the inference and learning equations in our models.

3.3 Learning in the state-observation model

So far, we assumed that the prediction and observation functions f and g as well as the associated covariance Σ_d and Σ_o were known. Usually, f and g are parametric functions defined by the physical properties of the system [1], and the noise amplitude for the observation and prediction processes can be found by a statistical analysis of the available data. This task is commonly referred to as *system identification*, and is sometimes followed by manual tuning. Learning the model parameters from data usually provides the best results, and proved to outperform manual optimization. More importantly, exploiting the learning machinery of the PGM framework allows for an accurate identification of arbitrarily complex distributions, that may also help in increasing the filter performance.

Multiple strategies can be chosen for learning the model parameters. It can either be done online, using sequential optimization steps, or offline, then based on a predefined training set (the batch method). The training set can also either contain partially observed data, usually corresponding to the observations $\{y_1, \ldots, y_t\}$, or fully observed data, in which case we

[1] Note that in the case of the LDS, f and g are fully represented by the matrices A and C.

require an accurate system for measuring the variables in x_t. Note that these measures are usually referred to as the *ground truth*. Finally, the model parameters can be trained in a generative approach or in a discriminative manner. The state-observation model being generative by construction, the most popular and intuitive approach is to find the parameters that explain at best the generative processes $p(x_t|x_{t-1})$ and $p(y_t|x_t)$ through maximum likelihood. However, the likelihood of the observed data in the training set does not measure the performance of the filter in estimating the posterior $p(x_t|y_1, \ldots, y_t)$. It is then reasonable to replace the model likelihood by an *error-driven* loss function in the parameter optimization process.

In their standard formulation, both generative and discriminative training are only applicable when the full state x_t is available. For many real systems, it is however more realistic to assume that we only observe a subset of the variables in x_t. For example, the state vector may describe both position and speed of the robot, while our accurate measurement system only provides the position values. Interestingly, we will see that the discriminative method can be straightforwardly adapted, regardless of the number of variables in the observed subset. Meanwhile, the Maximum likelihood approach is not always suitable since, when learning the model in a generative manner, we need accurate observations for all the variables in x_t that are actually exploited in the observation model[2].

Recall that when no ground truth is available, then the training set (thus formed of observations y_t only) must not contain frequent outlier occurrences or unreliable data. This is because in this configuration we can only rely on the filtering or smoothing capability of the model to reconstruct the sequence of hidden states (usually via the EM algorithm). Thus, parameter optimization is done based on some data that is likely to be incorrect. For this reason, we assume that an accurate measure of all or a subset of the variables in the state vector is provided for training. Logically, online parameter estimation is not considered in this work, as we consider that the availability of a form of ground truth is temporary, while online exploitation of our algorithms would require this information to be continuously available. Moreover, the permanent access to an accurate observation of the state would render the exploitation of Bayes filters irrelevant. In the next sections, we give more details about the generative and discriminative training approaches we shall exploit.

3.3.1 Generative training

In this section, we assume we are given a fully observed dataset $\mathcal{D} = \{\{x_1, y_1\}, \ldots, \{x_T, y_T\}\}$. Generative training then aims at maximizing the

[2]Note that it is very common for the observation model to ignore some components of the state.

joint likelihood of the sequence of states and associated observations $p(\mathcal{D}|\Theta)$ with respect to the model parameters Θ. Exploiting properly the d-separation properties of the model, the likelihood optimization breaks down into the independent optimization of two terms, which can be seen as the prediction likelihood and the observation likelihood:

$$p(\mathcal{D}|\Theta) = p(x_1|\mu_i, \Sigma_i) \prod_{t=2}^{T} p(x_t|x_{t-1}, \Theta_f, \Sigma_d) \prod_{t=1}^{T} p(y_t|x_t, \Theta_g, \Sigma_o) \qquad (3.5)$$

where Θ_f and Θ_g represent the prediction and observation function parameters. This is equivalent to solve a regression problem for the mapping $x_t \mapsto y_t$ and $x_{t-1} \mapsto x_t$. In practice, when the dataset is partially observed, this general idea of separately training the observation and prediction models is kept. The training method only differs by the exploitation of the EM algorithm in which smoothing is used to infer the hidden state variable within the E-step.

Since generative training is equivalent to regression, algorithms have been applied in order to build at first, parametric models (Bar-Shalom et al., 2002), and more recently non-parametric models (Deisenroth et al., 2009; Ko and Fox, 2009; Deisenroth et al., 2012). By learning an observation model without introducing any prior knowledge over the system behavior, the latter approach proved to be very efficient, and has been extended to a fully state-dependent observation model through the introduction of heteroscedastic noise model (meaning that the observation noise is state-dependent). This approach tries to compensate for the usual stationary noise (homoscedasticity) assumption made in traditional regression problems, that leads to the basic issues encountered in the presence of outliers. The resulting model is of the form $y_t = f(x_t) + \epsilon(x_t)$ where f and ϵ are represented by Gaussian processes (GP). Unfortunately, these models are still unable to properly distinguish which part of the observation signal should be assigned to the deterministic component f and which part should be considered as noise, especially in the case of strong and state-independent noise, *i.e* the typical unmodeled aspects of the observation process.

This problem is illustrated in Fig.3.3 where we trained an observation model with heteroscedastic noise following the method proposed in (Kersting et al., 2007). This example simulates the altitude measurement provided by an ultrasonic sensor presenting multiple outliers occurrences and a limited operation range of 6 meters. The model is trained on a first dataset. Then its prediction mean and variance (given x_t) are displayed concurrently with the measurements provided in a different test set. While this figure does not directly represent the impact of the exploitation of the model in a Bayes filter, it clearly shows that, as expected, the noise model is unable to provide consistent outputs as it is only able to reproduce the variation

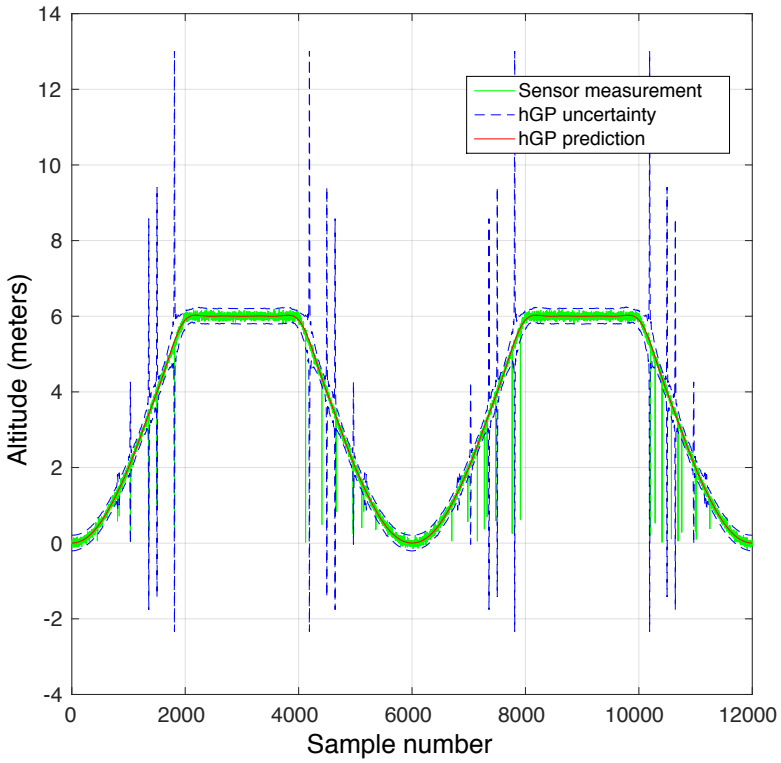


Figure 3.3: Output prediction of a GP-based heteroscedastic observation model trained on polluted data.

in the measurement accuracy that was observed in the training set. Consequently we can see that the uncertainty arbitrarily increases in some specific state values while the measurement is correct, whereas in many situations the measure is erroneous but the observation uncertainty stays low. While one must argue that the role of the noise model is not to represent strong perturbations in the measurement process, this example reveals that these methods are unable to deal with data prone to outliers, and as such must be exploited with great care.

3.3.2 Overcoming the regression issues with a new observation model

We have seen that general strategies as well as more advanced methods like the introduction of heteroscedasticity suffer strong shortcomings when trying to exploit polluted data. In this work, we suggest a different strategy by introducing a new method for modeling the observation distribution. This model is purposely built in order to improve its robustness with respect to measurement errors within the training set and later at runtime.

In our approach, the deterministic component g of the observation function[3] is assumed to be known, as it can be obtained through an analysis of the sensor physical properties. More precisely we assume that the deterministic mapping between x_t and each possible subset of measures within y_t is known. Based on this assumption, the remaining components of the model (and their optimization) are then fully dedicated to capturing the unmodeled aspects of the system. In other words we make a better exploitation of the available training data by pre-encoding what we already know about the system. While we do not learn the observation function itself, we however learn a selection component that acts as a selector over the individual measures within the observation vector y_t. There are two equivalent interpretations of this new component: it can either be considered as a measurement rejection (or selection) mechanism, or alternatively as a specific instantiation of adaptive filtering, where the model is continuously evolving. In this case the model evolution is confined to a finite set of possible observation functions that are selected online by an additional switch variable.

Similarly to heteroscedastic regression, we can also train a distinct model for continuously adapting the observation noise. Note that a common assumption made when learning the observation noise covariance is to require Σ_o to be diagonal. This assumption can usually be done without loss of generality and reduces the number of free parameters to learn while improving numerical stability (Murphy, 2012).

Finally recalling that our goal is to represent the context influence over the measurement generation process, the two selection and adaptation components are not state-dependent (as in classical heteroscedastic regression) but context-dependent. This prevents us from the issues arising when the observation distribution is learned through a pure regression problem $x_t \mapsto y_t$. The general principle of this new observation model is depicted in Fig.3.4.

3.3.3 Discriminative training

Maximum likelihood estimation, as exploited in generative training, is a popular method because it brings some convenient decomposition properties (Cf. eq.(2.7)), and more interestingly because it is a *consistent* estimator. Consistency means that if the model we chose is sufficiently expressive, then if the number of data instances in the training set goes to infinity, the maximum likelihood estimator converges to the true model (Koller and Friedman, 2009). In this case no other training criteria can provide better results. However, the model is rarely expressive enough, and thus maximum likelihood estimation may result in an irrelevant model. This problem is intensified by the specific nature of the likelihood score, which measures the strength of the dependencies between the variables and their parents

[3]Or equivalently the matrix C

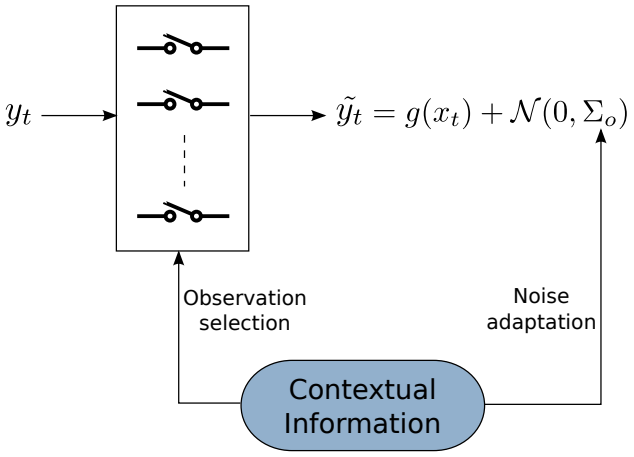


Figure 3.4: Basic principle of the robust and context-dependent observation distribution exploited in this work.

within the graph. Since likelihood maximization tends to fit precisely the specifics of the empirical distribution, *i.e* the training set, it undergoes the well-known overfitting issue.

Exploiting a generative model however does not necessarily mean that we are interested in the generative capabilities of the model. Recalling that our ultimate objective is to provide an accurate conditional probability over the state given the past observations, we intuitively understand that training the model without running inference (as done in generative training) does not allow us to properly assess a score relating the actual model performance. This is why we suggest an alternative loss function, which requires to run the recursive equations during the optimization process. We refer to this method as discriminative training, since we focus on the conditional output distribution over the states, and not the joint distribution over the states and observations.

Within the HMM community, discriminative methods are commonly used and known to outperform the maximum likelihood criterion, especially when incorrect modeling assumptions are made. Popular discriminative methods include *Viterbi training* (Allahverdyan and Galstyan, 2011), *Maximum mutual information* (Bahl et al., 1986) or more performance oriented methods such as *Minimum classification error* training (Juang and Katagiri, 1992). All these methods involve propagation of the hidden state variable through the model, instead of individual training of the observation and prediction distribution. However, these algorithm are especially designed for HMMs and can not be directly exploited in the continuous case.

Recall that a strict formalisation of discriminative training over a state-space model requires to maximize the conditional likelihood of the training

set $p(x_1, \ldots, x_T | y_1, \ldots, y_T)$. Unlike joint likelihood this conditional distribution does not decompose into simple terms. Consequently we 'force' the decomposition by replacing the discriminative likelihood by an alternative objective function, defined as a product of subproblems that directly express the filter performance in terms of state accuracy and uncertainty. An intuitive and simple score of the filter performance is the local posterior distribution $p(x_t | y_1, \ldots, y_t)$ which corresponds to the output distribution of the recursive equation (3.1) evaluated at the training sample $\{x_t, y_t\}$. The resulting objective function was also exploited in (Abbeel et al., 2005) and can be analogously seen as the minimum classification error criterion applied to the state estimation problem. The discriminative likelihood is then defined as:

$$\mathcal{L}_{discr} = \sum_{t=1}^{T} log\big(p(x_t | y_1, \ldots, y_t) \big) \tag{3.6}$$

Exploiting incomplete state observations:

An interesting property brought by this alternative loss function is that it is straightforwardly exploitable if we are only provided with observations of a subset of the variables in x_t. For example, if the training set $\mathcal{D}$ contains measurement pairs $\{v_t, y_t\}$ so that

$$v_t = h(x_t) + \mathcal{N}(O, \Sigma_h)$$

where v_t is the vector of observed variables in x_t provided by our accurate measurement system, h is a projective function acting as a selector over the variables in x_t and Σ_h is the (low) observation variance of the measurement system used to capture a subset of the hidden state variable.

Then we can optimize $\mathcal{L}_{discr}$ so as to explain as best the observations in the training set $\mathcal{D} = \{\{v_1, y_1\}, \ldots, \{v_T, y_T\}\}$ *i.e*:

$$\mathcal{L}_{discr} = \sum_{t=1}^{T} log\big(p(v_t | y_1, \ldots, y_t)\big) \tag{3.7}$$

noting that we have

$$p(v_t | y_1, \ldots, y_t) = \int p(v_t | x_t) p(x_t | y_1, \ldots, y_t) \, \mathrm{d}x_t$$

The projection h being equivalent to the product of a selection matrix H with the hidden state vector x_t, we can exploit once again the conjugate property of the linear Gaussian model to evaluate the marginal distribution over v_t using (A.1). This gives:

$$p(v_t | y_1, \ldots, y_t) = \mathcal{N}(v_t | H\mu_t, \Sigma_h + HV_tH^t) \tag{3.8}$$

Training can then be done using the coordinate ascent algorithm to find estimates for the unknown parameters of the observation distribution. While independent optimization of each local term within (3.6) converges to a global maximum, the optimization of the complete sum is however much more complex. This is because each posterior $p(x_t|y_1, \ldots, y_t)$ directly depends on the past estimates, and thus global optimization of the sum is equivalent to finding the optimal parameters that correspond to the sequence $x_1, \ldots, x_t$ (or equivalently $v_1, \ldots, v_t$). Consequently, the optimization of (3.6) is likely to converge to a local optimum, at the risk of locally improving strong errors in the state estimate while neglecting lower errors produced at other sample pairs $\{x_t, y_t\}$. In order to avoid to converge to such faulty local optimum, the discriminative loss function can be initialized with the parameters provided by a previous generative training step.

3.3.4 Discriminative training, or discriminative model ?

Until now, and as it is commonly done in the literature, we referred to the optimization of an error-driven objective function as discriminative training. In practice, the discriminative criterion we introduced might seem arbitrary, albeit intuitive. This is because it does not follow a basic rule that we described in chapter 2: a model encodes a single joint distribution over its variables. For instance, a state-observation model encodes the joint likelihood of the state and the observations (3.5). Exploiting a different likelihood for learning the parameters consequently means that we arbitrarily reinterpret the model. We now provide an alternative viewpoint for this approach, in order to provide a more consistent formalization of the problem regarding the probabilistic framework.

As suggested in (Minka, 2005; Bishop and Lasserre, 2007), referring to distinct generative and discriminative training methods in a generative model[4] is highly debatable. In fact, when we perform discriminative training on a generative model, we change the meaning of the model and implicitly build an equivalent discriminative model. The (conditional) likelihood optimization of this new model corresponds to the discriminative training of the initial generative model. For more consistency with the probabilistic framework, we should preferably refer to generative and discriminative models. And training then always consists in optimizing the joint distribution encoded by the corresponding generative or discriminative model.

Let us reconsider the 'discriminative' likelihood $p(x_1, \ldots, x_T|y_1, \ldots, y_T)$. As mentioned earlier, this distribution does not easily decompose in convenient[5] terms if we exploit the generative state-observation model. Especially, the d-separation property of the model does not yield the likelihood to be

[4]Recall that a discriminative model can only be 'discriminatively' trained

[5]For the optimization task

decomposed in a product of posterior state distributions $p(x_t|y_1, \ldots, y_t)$. However, if we now introduce the corresponding discriminative model as depicted in Fig.3.5, and recalling that we have $\mathcal{D} = \{\{x_1, y_1\}, \ldots, \{x_T, y_T\}\}$, we can decompose the joint likelihood of the model as:

$$p(\mathcal{D}) = p(x_1|y_1) \prod_{t=2}^{T} p(x_t|x_{t-1}, y_t) \tag{3.9}$$

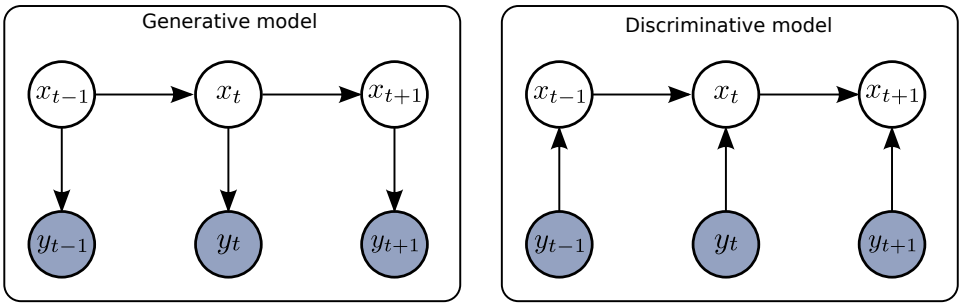


Figure 3.5: Original generative state-observation model (on the left), and its corresponding discriminative representation (on the right).

In this new discriminative model, we see that the posterior belief over the state x_t depends on the prior state x_{t-1} and the current observation y_t. For clarity, this distribution can be represented through the graphical model depicted in Fig.3.6 where the variable x_{t-1} is preferably represented by a set of static prior parameters corresponding to the propagation of x_{t-1} through the prediction model. This gives us a prior mean and covariance $\mu_{t|t-1} = A\mu_{t-1}$ and $V_{t|t-1} = P_{t-1}$ provided by eq.(3.3). Since we only defined an observation model $p(y_t|x_t)$, the posterior $p(x_t|x_{t-1}, y_t)$ is obtained by exploiting the property (A.2) where the prior over x_t is given by $\mu_{t|t-1}$ and $V_{t|t-1}$. This operation corresponds to the evaluation of the contrapositive distribution $p(x_t|y_t)$ as done in (3.4), an thus leads to the Kalman equations. Finally, we see that the likelihood encoded by the new model corresponds to the product of posterior distributions as defined in (3.6).

Interestingly, this equivalent representation is analogous to the *Maximum Entropy Markov Model* introduced in (McCallum et al., 2000), except that the state variable is here continuous, and the posterior state distribution computation still relies on a generative observation model. In other words we exploit a discriminative version of the state-observation model where the prediction and observation components are still generative representations. The resulting model can however be seen as a discriminative predictor $x_{t-1}, y_t \mapsto x_t$.

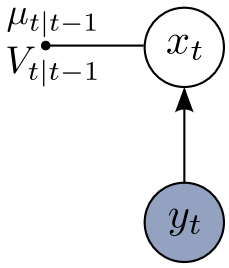


Figure 3.6: Equivalent representation of the distribution $p(x_t|x_{t-1}, y_t)$. The prior state variable x_{t-1} is replaced by the resulting predicted mean and variance after propagation in the prediction model. Thus it is not represented by a random variable but by static parameters.

Decomposing the evaluation of the distribution $p(x_t|x_{t-1}, y_t)$ in two separates steps involving the prior information over x_{t-1} and the observation y_t can be seen as a consequence of the model structure. Indeed, in this new model the observation y_t does not influence the states prior to time t, due to the inactive trail $x_{t-1} \rightarrow x_t \leftarrow y_t$. Consequently, the effect of the prior information can be taken into account separately, before exploiting the information provided by the observation. An other consequence of this independence property is that the model does not allow to perform smoothing. This can be an issue in some applications, but not in our case since we only aim at performing filtering.

Note that the debate about discriminative or generative modeling is still ongoing, and examples of the proper exploitation of the dual representation of an equivalent generative and discriminative model stays rare. In this work, we try to keep a consistent understanding of the model we exploit regarding the PGM framework, and we consequently prefer to clarify why the exploitation of discriminative likelihood can be explained within the same, unifying framework. To our knowledge, this re-interpretation of discriminative training for state-observation models is new, and as such should be considered as a theoretical contribution. Note that, for simplicity, we may sometimes still refer to generative and discriminative training throughout this manuscript despite the inaccuracy of this denomination.

3.4 Conclusion

In this chapter we detailed the classical filtering algorithm in the state-space model, as well as the prevalent approaches for learning the prediction and observation distributions. We suggested a different paradigm for building and learning a robust observation distribution that can deal with frequent occurrence of strong outliers. In practice, this model differs from traditional approaches in two points. First, it considers that a rejection (or selection)

mechanism has to be explicitly introduced in order to explain the mapping $x_t \mapsto y_t$. Second, it is accepted that the information provided by x_t is not sufficient to explain the measurement generation process and the resulting observation distribution is consequently context-dependent. Inspired by heteroscedastic regression models, the selection mechanism is completed by a noise adaptation component that is also context-dependent.

We also detailed an alternative optimization criterion for learning the model parameters that can be interpreted as re-modeling our system. This is deeply linked to the current discussion of issues concerning the discriminative training of generative models. In this debate, the consensus seems to be that there are only generative and discriminative models, and training is always consistently done through the joint likelihood optimization.

In the next chapters we will provide details about some interesting families of models that can be exploited for the selection and adaptation task. Since it slightly differs from the usual state-observation based implementation, the proposed observation model also differs in the training methods that have to be used. These training methods will be detailed along with each suggested model.

Chapter 4

Context-dependent measurement selection: An approach based on the Mixture of Experts

Contents

In this chapter, we provide a first solution for context-dependent measurement selection. This task can be considered to be close to outlier rejection, except that we never properly characterize measurements by their relative distance to a density function. Instead, we focus on the utility of each measurement in improving the state estimate. This knowledge is encoded in a new component that selects at each time step the best measurement subset. Consequently, the proposed solution fundamentally differs from the existing approaches that fully rely on the prior knowledge encoded through the state-observation model. As described in the next sections, we should see these developments as an extension of the multiple model approach. This chapter presents in details the work which was published in (Ravet et al., 2013).

4.1 Motivation - beyond measurement rejection

The issue of understanding and exploiting measurements provided by different sensors is of major importance in state estimation. In practice, one has to deal with a whole range of performance alterations going from the unavoidable average observation noise to completely unreliable measure values. The usual approach taken to cope with unmodeled observation alterations is outlier rejection, which consists in rejecting any measurement lying some distance away from the expected observation distribution $p(y_t|x_t)$. The outlier rejection method is strongly entrenched in the domain of Bayesian filtering, and relies on the strong assumption that we are given a sound prior knowledge about the observation distribution, so that we can legitimately reject any measurement that lies too far from its predicted distribution. Multiple classes of methods for outlier rejection have been developed, and most of them rely on manual parameter tuning, rejection heuristics, or heavy computations that leads the filter to be unsuitable for real-time applications (Kitagawa, 1987; 1996; Gordon and Smith, 1993). Recent work has however shown interesting solutions for learning online the parameters of a robust Kalman filter that is able to process corrupted data (Ting et al., 2007; Agamennoni et al., 2011).

Alternatively, the existence of transient disturbances can be directly modeled within the same complex observation distribution, which now explicitly represents some identified cases of measurement error. An illustration of this method is the well known beam model defined for a laser telemeter in (Thrun et al., 2005b). As depicted in Fig. 4.1, such models rely on a prior knowledge about the different phenomenons that may lead to erroneous measurements.

This approach then requires a deep understanding of the sensor behavior and introduces a subjective representation of the different types of errors that can occur through an associated local distribution. It also makes in-

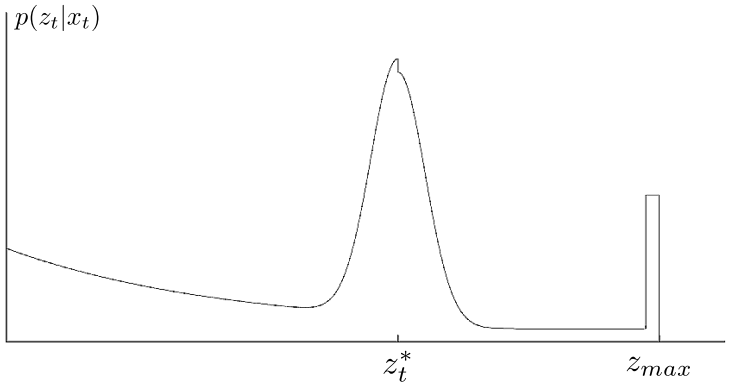


Figure 4.1: The beam model pseudo-density representing the distance z_t measured by a laser beam given the robot position x_t. $p(z_t|x_t)$ is defined as a mixture of distributions representing different types of situations: unexpected object shortening the perceived range in the first part, correct measurement with Gaussian noise in the center (z_t^*), and max-range measurement. (Excerpt from (Thrun et al., 2005b) p.128)

ference in the state-observation model more complex as there is usually no closed-form expression for the recursive equations. Thus one has to turn to sampling methods which come at increased computational costs and introduce approximations in the state estimate.

Whether we explicitly represent the different types of alterations or not, these approaches may perform poorly for a common reason: observations are treated within a self-contained system. In this context, we have no guarantee that there is no combination of measurement that can mislead the system. Consequently, we are never sure to give an adequate treatment to the current observation. In other words, while these models encode the existence of alterations, and sometimes depict them with a dedicated model, they are unable to capture the actual occurrence of the phenomenon underlying the potential measurement alteration. Instead, they can only try to infer the current perception context based on self-consistency checks between the real observation and the model.

Our objective is to go beyond the traditional rejection methods. In this context we shall see our task as optimal exploitation of the available measurements, following time-varying decisions rules. For this reason, our task is preferably referred to as measurement selection, since we do not aim solely at rejecting the measurements that do not fit a stationary model designed after a prior knowledge. In practice, by selecting the subset of measurements that results in the best state estimate, we implicitly perform rejection. The issue of robustness is then automatically encompassed within a

more general objective: providing the best state estimate given the available measurements.

4.2 Approach

By artificially introducing multiple filters exploiting different measurement subsets within the observation vector y_t, we can map the measurement selection problem to the *multiple model* approach. The multiple model approach can be seen as a discrete solution to the problem of adaptive filtering. Instead of continuously adjusting the filter parameters, a set of distinct models are initially defined, and assumed to be optimally parametrized for the specific regimes of the system. Then the multiple model approach proceeds by switching at runtime between the different models in order to continuously provide the optimal output. For this reason, this scheme is usually referred to as the multiple model adaptive estimator (MMAE). Originally, this principle was applied to target tracking, in order to model the different regimes, or motion patterns of the target (Bar-Shalom et al., 2002). In our case, the multiple models differ only in the observation distributions, acting as a selector on the measurements. This approach is depicted in Fig. 4.2.

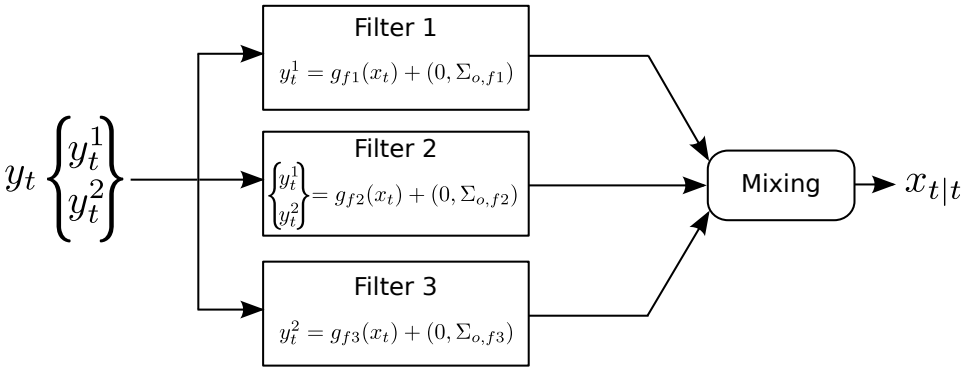


Figure 4.2: The multiple model approach applied to measurement selection. Illustration with a 2-dimensional observation.

In the next sections, we give a background of the existing multiple model approaches and suggest an alternative solution, allowing to easily introduce an additional input variable (the context) within the model.

4.2.1 Background: The multiple model approach

In a Bayesian representation, the multiple model approach assumes that the changing behavior of the system can be represented by an additional discrete hidden variable $s_t \in \{1, \ldots, K\}$ that governs the parameters of a

classical state-observation model, the parameters taking values in a predefined discrete set. The evolution of s_t is modeled through an independent HMM and either the observation distribution, the prediction distribution, or both of them now vary with the switching discrete state. This configuration is described in Fig. 4.3. For this reason, these approaches are referred to as *swithing state-space model* (SSSM), or *hybrid discrete/continuous* state space model.

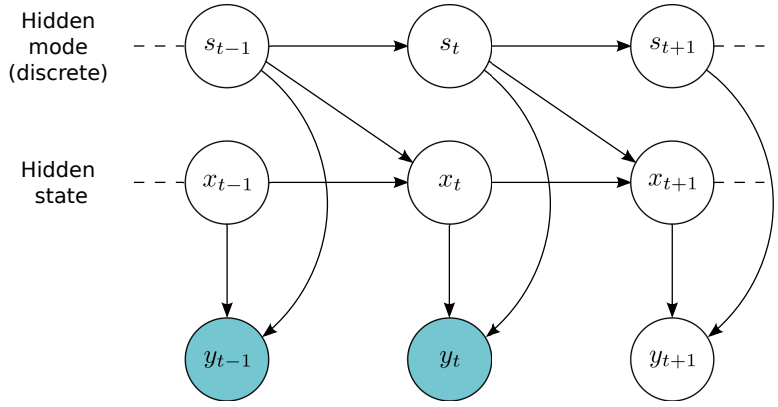


Figure 4.3: Switching state space model where both the observation and prediction distributions can switch with the hidden state.

Due to the multiple hypotheses induced by the hidden discrete state, exact inference on SSSMs is intractable. This is because the first belief state is described by a mixture of K distributions, each of these distributions then being used to compute the next state value for each possible model. Hence, the exact mixture distribution representing the state belief exponentially increases in time, and must therefore be approximated. Different approximation methods can be found in the literature, according to the nature of the filter. When each filter is a LDS, some deterministic approximations can be performed. A popular approximation method for combining the multiple distributions is the generalized pseudo-Bayes approach (GPB). In the first-order generalized pseudo-Bayes (GPB1), the resulting K distributions are combined within a single component at each time step, while for the second-order generalized pseudo-Bayes (GPB2), the previous K distributions are propagated through each model before being combined. Thus these methods respectively require to run in parallel K and K^2 filters at each iteration. The GPB method was improved through the interacting multiple model (IMM) approach (Blom and Bar-Shalom, 1988) which received a lot of attention as it provides a similar approximation to GPB2 but only requires to run K filters in parallel.

To see why these approaches are not a satisfying solution, we shall derive

the inference equation for the discrete hidden state s_t. This requires to evaluate $P(s_t = j|y_{1:t})$, where we start by exploiting the Markov chain formed by s_t and s_{t-1}. For any $i, j \in \{1, \ldots, K\}$, we have:

$$P(s_t = j|y_{1:t}) = \sum_i P(s_{t-1} = i, s_t = j|y_{1:t})$$

Then using Bayes theorem we can write:

$$\begin{aligned}
&P(s_{t-1} = i, s_t = j|y_{1:t-1}, y_t) \\
&= \frac{1}{c} P(y_t|s_{t-1} = i, s_t = j, y_{1:t-1}) P(s_{t-1} = i, s_t = j|y_{1:t-1}) \\
&= \frac{1}{c} P(y_t|s_{t-1} = i, s_t = j, y_{1:t-1}) P(s_t = j|s_{t-1} = i, y_{1:t-1}) P(s_{t-1} = i|y_{1:t-1}) \\
&= \frac{1}{c} L_t(i, j) Z(i, j) P(s_{t-1} = i|y_{1:t-1})
\end{aligned}$$

where

$$c = \sum_i \sum_j L_t(i, j) Z(i, j) P(s_{t-1} = i|y_{1:t-1})$$

with $L_t(i, j)$ the likelihood of the observation y_t in the case $S_t = j$ and $S_{t-1} = i$ and $Z(i, j)$ the element (i, j) in the homogeneous transition matrix defined for the Markov chain over the discrete state. Consequently, we see that the belief of the current mode still relies on self-consistency of the observation with the prior knowledge introduced through the different models and the fixed transition rules between the different modes. Furthermore, the switches between the different modes are homogeneous Markov, a very strong assumption if we aim at modeling the context evolution.

Although some authors augmented the IMM with context-dependent information (Schubert and Wanielik, 2009) through the introduction of an additional Bayesian network governing the transition probabilities, this model fundamentally relies on the exploitation of internal estimates and predefined transition matrices. An analogous approach can be found in (de Freitas et al., 2004) in the context of fault diagnosis. Based on a jump Markov linear model, this method requires to know the different regimes of operation for the learning step, while we want our system to discover these different contexts by itself.

An exception to the previously discussed models can be found in (Bengio and Frasconi, 1996) where the transition and observation distributions of an HMM are conditionally dependent on a new input variable. This model is known as *input-output HMM* (IOHMM) and only considers the case of discrete state dynamical systems. Note that the *maximum entropy Markov model* developed in (McCallum et al., 2000) (MEMM) can be seen as a special instance of IOHMM as it depicts a Markov chain in which the transition probabilities depend on a set of input features. Interestingly, this model is

more likely to be considered as a special instance of Conditional Random Field, and as such requires to design by hand or to train independently the feature functions describing the relationship between the state and the observation.

Whether they are input-dependent or not, these models however assume that there exists a sequential pattern underlying the switching behavior of the real system. This is because they were mainly designed for sequential data labelling. As explained before, we are not willing to introduce any temporal aspect when modeling the context influence, as we prefer to switch arbitrarily between different regimes. For this purpose, we explore a different switching model that was not originally designed for modeling dynamic systems, which is known as the Mixture of Experts model.

4.2.2 Background: The mixture of experts model

Aiming at learning how to combine some complementary experts, the mixture of experts model (ME) framework lends itself very well to the model selection problem as it basically computes an optimal output through a weighted sum of individual experts. To achieve this mediation task, the ME relies on a gating network in charge of providing gating probabilities, equivalent to reliability coefficients over the set of experts. When experts are replaced by Bayes filters, this approach provides an efficient alternative to the multiple model approach (Chaer et al., 1998) (Chaer et al., 1997).

The mixture of experts approach basically consists in decomposing a complex problem into subtasks, each of which being handled by an appropriate expert. Traditionally used for regression or classification problems, the model learns to split the input space into overlapping regions within which assigned experts are active.

The standard ME framework (Jacobs et al., 1991) consists in a set of K experts modules and a gating network (See Fig. 4.4). Each expert $k = 1...K$ associated with parameters λ_k looks at the input vector y and computes a local output x_k through a function $f_k(\lambda_k, y)$. In a probabilistic interpretation, the output of an expert k can be viewed as the mean of a probability distribution $P(x|y, \lambda_k)$ with x the desired target value associated to the input y. Assuming that the different experts may be more competent in different regions of the input space (*i.e.* they have higher probability to produce the desired target x), the gating network mediates the outputs of the bank of experts. For this purpose the gating network produces for each expert k a probability of its output x_k to be equal to the desired output x. This result in a set of gating probabilities g_k weighting the output of all experts while satisfying constraints $g_k \geq 0, k = 1...K$, and $\sum_{k=1}^{K} g_k = 1$.

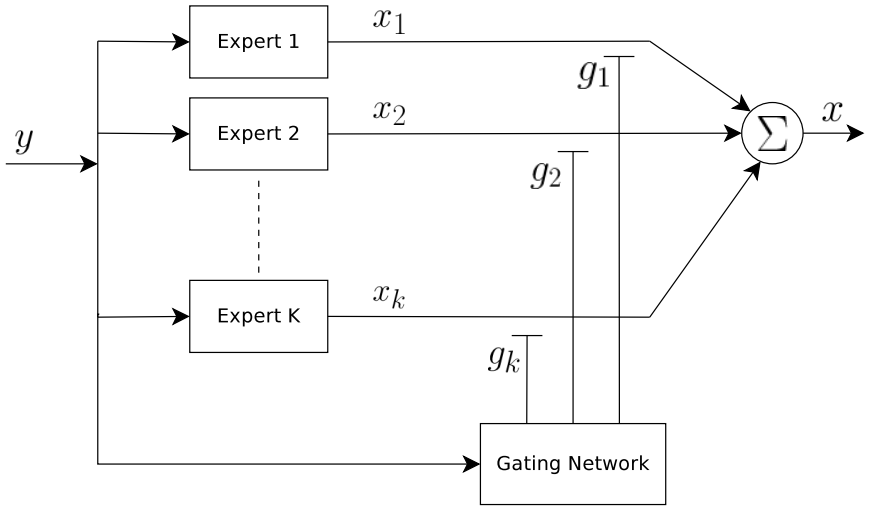


Figure 4.4: The basic mixture of experts framework.

Given an input vector y and a target vector x, the probability of observing x is then written in terms of gating probabilities and experts outputs (using product rule) as

$$\begin{aligned} P(x|y,\Theta,\Lambda) &= \sum_{k=1}^{K} P(x,k|y,\Theta,\Lambda) \\ &= \sum_{k=1}^{K} P(k|y,\Theta)P(x|k,y,\Lambda) \\ &= \sum_{k=1}^{K} g_k(y,\theta_k)P(x|y,\lambda_k) \end{aligned} \tag{4.1}$$

where g_k is the weight for expert k and $\{\Theta,\Lambda\}$ denotes the set of all parameters, with $\Theta = \{\theta_k\}_{k=1}^{K}$ the set of gating parameters and $\Lambda = \{\lambda_k\}_{k=1}^{K}$ the set of experts parameters.

ME implementations then differ in 3 main points: the experts model, the gating model, and the inference method. More information about the different implementations can be found in (Yuksel et al., 2012).

Given a training set $\{\mathbf{x},\mathbf{y}\} = \big\{\{x_1,y_1\},\ldots,\{x_N,y_N\}\big\}$ we try to maximize the likelihood $\mathcal{L}$ of the data set with respect to the model parameters. If samples are considered identically independently distributed, this is equivalent to maximize:

$$\mathcal{L} = \prod_n p(x_n,y_n)$$

We then define the usual cost function $\mathcal{C}$ as the negative log of the likelihood

function, so that maximizing the likelihood is equivalent to minimize $\mathcal{C}$:

$$\mathcal{C} = -\sum_{n} ln(p(x_n, y_n))$$

Different methods for determining the max likelihood have been developed. The standard gradient descent methods can be applied. More recently, sampling, variational inference and several Expectation Maximization (EM) algorithms have emerged (Yuksel et al., 2012) and have shown good performances.

The original Mixture of Experts model

In the standard model (Jacobs et al., 1991) the experts networks are single layer linear networks – which means $f_k(\lambda_k, y) = \lambda_k^T y$. The expert conditional density function is considered to be a d-dimensional Gaussian distribution (for a d-dimensional vector x):

$$P(x|k, y, \lambda_k) = \frac{1}{(2\pi)^{d/2}} exp(-\frac{\parallel x - f_k(\lambda_k, y) \parallel^2}{2})$$

And the gating network is a single layer network with soft-max activation function, so that:

$$g_k = \frac{exp(u_k)}{\sum_{j=1}^{K} exp(u_j)}$$

where $u_k = \theta_k^T y$ are the gating network outputs before normalization. Here the soft-max function ensures that the gating probabilities sum to unity and are non-negative. This model is known as linear logistic and implicitly induces competitiveness between the experts.

One strong limitation of this global gating network is caused by the single layer linear implementation. This model divides the input space into overlapping regions (Ramamurti and Ghosh, 1998) with soft hyperplanes (meaning that data samples close to decision boundaries can lie simultaneously in different regions) and assign one or a combination of experts to this region. Intuitively we can understand that these specific decision boundaries will also cause interference among experts where the half-subspaces are overlapping. These interferences can happen to be constructive (or locally constructive), improving performance of a single expert, however this is more likely to cause wrong reconstruction, especially when the number of experts increases. Note that because of the nonlinearity of the soft-max function, there is no analytic solution for the optimal parameters of the gating network when maximizing the model likelihood. Hence, learning requires to turn to more complex methods like the iterative reweighted least squares algorithm (Jordan, 1993).

To circumvent the problem of overlapping sub-spaces, some authors (Jordan, 1993) extended the model to a more complex architecture, called Hierarchical Mixture of Experts (HME). This model basically consists in introducing extra levels of mixing layers, by creating subsets of individual experts providing first level output through a local gating network. These first-level outputs are then mixed by a high-level gating network. The hierarchy among subsets of experts provides isolation of sub-spaces, but also increases computational cost as the tree height increases (more experts and gating networks need to be learned). During training one also has to deal with the complex problem of defining the appropriate depth of the tree.

For this reason, we prefer to apply the single gating level structure, but switch to an other gating network model, known as *localized gating network*, which provides a better modeling flexibility.

Mixture of Experts with localized gating network

Given the previously discussed issues when exploiting a single layer gating network, we adopt a different model allowing to assign one expert to one precise region of the input space. This modified gating network is called the localized ME (Xu et al., 1995) and consists of normalized Gaussian kernels (or any density function from the exponential family):

$$g_k(y, \theta_k) = P(k|y) = \frac{\alpha_k P(y|\theta_k)}{\sum_{j=1}^{K} \alpha_j P(y|\theta_j)} \tag{4.2}$$

with

$$P(y|\theta_k) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_k \mid^{1/2}} exp\left(-\frac{(y - m_k)^T \Sigma_k^{-1} (y - m_k)}{2} \right)$$

where $\theta_k = \{m_k, \Sigma_k\}$ defines the mean and variance of the Gaussian kernel distribution and d the dimension of the vector y.

The Gaussian kernels allow to divide the input space into soft hyper-ellipsoids. These ellipsoids can overlap, or create localized regions of expertise where a single sensor is trustworthy. Gausian kernels also simplify the learning step, as it conducts to a one-pass maximization step when using the EM algorithm. The convergence rate of the EM algorithm is also empirically and theoretically proven to be faster than gradient ascent methods (Jordan and Xu, 1993). Associated to the Gaussian kernels, it provides guaranteed convergence due to the single loop maximization step (as opposed to the method developed in (Jordan, 1993) which consists in a double-loop EM). For this reason we decide to learn the gating parameters with the EM algorithm.

Training the localized ME with Expectation Maximization:

The basic idea of the EM algorithm is to make the assumption that some variables are hidden, in our case the probability that the n^{th} target sample x_n was generated by expert k. Hence we introduce an indicator variable z :

$$z_n^j = \begin{cases} 1 & \text{if target sample } x_n \text{ is generated by expert } j \\ 0 & \text{otherwise} \end{cases}$$

This hidden variable induces mutual competition among experts. It also models the existence of unknown operating contexts which for different subsets of experts are reliable. To obtain a one pass calculation for the gating parameters, we perform maximum likelihood estimation on the joint density $p(x, y)$ (Xu et al., 1995). Denoting the k^{th} expert output conditional density function by $\phi_k(x|y)$, and using the new gating function, we rewrite equation (4.1) to get:

$$p(x|y, \Theta, \Lambda) = \sum_{k=1}^{K} \frac{\alpha_k P(y|\theta_k)}{\sum_{j=1}^{K} \alpha_j P(y|\theta_j)} \phi_k(x|y) \tag{4.3}$$

Using Baye's rule on equation (4.2) to obtain $p(y) = \sum_{j=1}^{K} \alpha_j P(y|\theta_j)$, we obtain the joint density

$$p(x, y|, \Theta, \Lambda) = \sum_{k=1}^{K} \alpha_k P(y|\theta_k) \phi_k(x|y) \tag{4.4}$$

Finally, introducing the indicator variable z to mediate mutually exclusive experts, the joint distribution over hidden and observed variables take the form

$$p(x, y, z) = \prod_{k=1}^{K} (\alpha_k P(y|\theta_k) \phi_k(x|y))^{z^k} \tag{4.5}$$

which by maximum likelihood leads to the cost function:

$$\mathcal{C} = -\sum_{n} \sum_{k=1}^{K} z_n^k \, ln(\alpha_k \, P(y_n|\theta_k) \phi_k(x_n|y_n)) \tag{4.6}$$

Now the specificity of EM algorithm comes into play. In the first step we replace the hidden variable z by its expected value. This is the expectation step.

Expectation:

$$\begin{aligned}\mathcal{E}(z_n^k) &:= p(z_n^k = 1|x_n, y_n) \\ &= \frac{p(x_n|z_n^k = 1, y_n)p(z_n^k = 1|y_n)}{p(x_n|y_n)} \\ &= \frac{\alpha_k P(y_n|\theta_k)\phi_k(x_n|y_n)}{\sum_{j=1}^{K} \alpha_j P(y_n|\theta_j)\phi_j(x_n|y_n)} = h_k(x_n, y_n) \end{aligned} \tag{4.7}$$

Then we maximize the expectation of the cost function by substituting z_k by its expectation $h_k(x, y)$. This is the maximization step.

Maximization:

$$\begin{aligned}\mathcal{E}(\mathcal{C}) &= -\sum_n \sum_{k=1}^{K} h_k(x_n, y_n) ln\big(\alpha_k \, P(y_n|\theta_k)\phi_k(x_n|y_n)\big) \\ &= -\sum_n \sum_{k=1}^{K} h_k(x_n, y_n) ln(\alpha_k P(y_n|\theta_k)) \\ &\quad -\sum_n \sum_{k=1}^{K} h_k(x_n, y_n) ln(\phi_k(x_n|y_n)) \end{aligned} \tag{4.8}$$

As we can see this cost function can be separated in two terms. The first one corresponds to the cost function relative to gating parameters ($\alpha_k P(y_n|\theta_k)$) and the second term corresponds to the expert network parameters.

4.3 Context-dependent measurement selection with the mixture of experts model

4.3.1 Approach

The powerful mixture of experts model is now exploited for measurement selection, following the multiple model approach discussed in section 4.2. In our context each expert is replaced by an individual Kalman filter providing its own estimation based on a subset of measurements $\{y_n^k\} \subseteq y_n$ and on its parameters λ_k describing the corresponding measurements noise and observation selection matrix. Note that the number and nature of measurement subsets we decide to exploit is problem dependent and may require some expertise. However it is always possible to use the combinatorially exhaustive number of subsets.

Consequently, we can represent the resulting model by a bank of filter running in parallel, differing in their observation functions, and whose output estimates are weighted by the gating network. This principle is depicted in Fig. 4.5 where we show that the gating network can share some filters inputs and exploit specific context information as well.

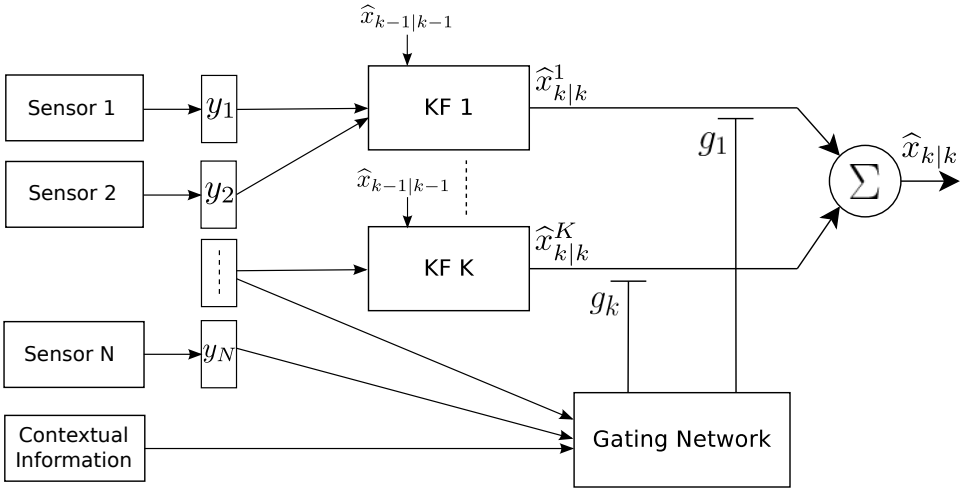


Figure 4.5: Mixture of experts framework for sensor selection. Each filter can be wired to a different subset of sensors. The gating network can share experts inputs and use any useful additional information for assessing the context.

At first, we will consider that the model parameters are known and discuss the inference task which, as for any multiple model approach requires approximation. Then, training of the gating network will be detailed, based on the work from Xu (Xu et al., 1995). In the following, we will assume that the different state-observation models (the multiple models) are linear and Gaussian. This assumption is made for the sake of clarity while conducting the theoretical analysis of the approach. However, non-linear models can be equally used within the same framework through simple adaptation steps which are described later.

4.3.2 Inference

As explained previously, one unavoidable issue with the multiple model approach is that the exact belief state grows exponentially in time. This is because for a set of K filters, each iteration produces K independent state estimates and each of these possible beliefs have to be considered in the prediction step of the subsequent iteration. Consequently, at iteration $t = T$, the exact distribution of the state is a mixture of K^T Gaussian distributions. To deal with this exponential growth we approximate the resulting posteri-

ors by a single convenient distribution, an approach typically referred to as *assumed density filtering*. We consequently use the GPB collapsing method of order 1 (GPB1), and approximate the mixture of filters output distribution with a single Gaussian distribution. At each step, if each filter k provides an output distribution of mean μ_k and variance σ_k with a corresponding gating weight g_k, we obtain the mixture distribution mean μ_{mix} and variance σ_{mix} (Bar-Shalom et al., 2002):

$$\mu_{mix} = \sum_{k=1}^{K} g_k \mu_k$$

$$\sigma_{mix} = \sum_{k=1}^{K} g_k [\sigma_k + (\mu_k - \mu_{mix})(\mu_k - \mu_{mix})^T]$$

The next transition step is then based on this mixture output, hence accumulating the error introduced by the approximation at each time step. However, it has been shown in (Boyen and Koller, 1998) that the process error remains bounded indefinitely, avoiding the mixture output to become irrelevant.

Alternatively, instead of approximating the posteriors by a single distribution, one solution consists in pruning the less probable estimates at each iteration. Note that this approach has not been tested yet.

4.3.3 Learning

By focusing on different objectives, we can alternatively exploit generative or discriminative training to find the gating network parameters. Note that we assume here that the noise model for each measurement within y_n is known and stationary, some methods for learning context-dependent noise models being provided later.

Discriminative training

In discriminative training, we aim at optimizing the gating network parameters with respect to the resulting state accuracy. Hence $\phi_k(x_n|y_n)$ is obtained by evaluating the output distribution of the k^{th} Kalman filter for the sample value x_n. Note that this fully exploits the discriminative equivalent model introduced in chapter 3 in which we see each filter as a discriminative predictor $y_n \mapsto x_n$. The model also differs from the basic mixture of experts in that the gating network uses a different input variable c corresponding to the context variable. The difference between these two models is illustrated in Fig. 4.6.

Given the two input vectors y and c, we now rewrite the basic prediction equation (4.1) for the new context-dependent mixture model:

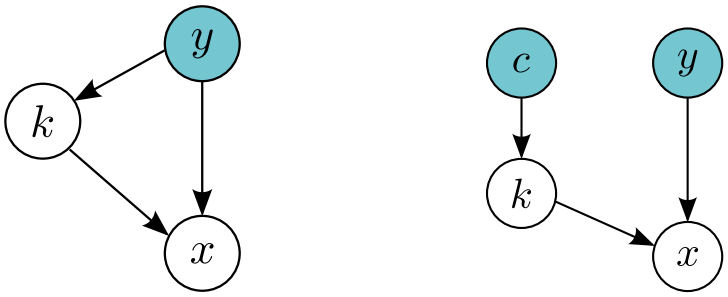


Figure 4.6: Graphical model for the basic Mixture of experts model (left), and the Mixture of experts applied to context-dependent observation distribution (right).

$$
\begin{aligned}
P(x|y,c,\Theta,\Lambda) &= \sum_{k=1}^{K} P(x,k|y,c,\Theta,\Lambda) \\
&= \sum_{k=1}^{K} P(k|y,c,\Theta)P(x|k,y,c,\Lambda) \\
&= \sum_{k=1}^{K} g_k(c,\theta_k)P(x|y,\lambda_k) \\
&= \sum_{k=1}^{K} g_k(c,\theta_k)\phi_k(x|y) \qquad (4.9)
\end{aligned}
$$

where we note that x is independent of c given k (using d-separation) so that $P(x|k,y,c,\Lambda) = P(x|y,\lambda_k)$. Similarly, we exploited the inactive trail $k \rightarrow x \leftarrow y$ to show that k is independent of y and that we have $P(k|y,c,\Theta) = P(k|c,\Theta)$.

We now introduce the localized gating network to rewrite (4.9):

$$P(x|y,c,\Theta,\Lambda) = \sum_{k=1}^{K} \frac{\alpha_k P(c|\theta_k)}{\sum_{j=1}^{K} \alpha_j P(c|\theta_j)} \phi_k(x|y) \qquad (4.10)$$

and noting that we have $p(c) = \sum_{j=1}^{K} \alpha_j P(c|\theta_j)$, we obtain the conditional density :

$$p(x,c|y,\Theta,\Lambda) = \sum_{k=1}^{K} \alpha_k P(c|\theta_k)\phi_k(x|y) \qquad (4.11)$$

Reintroducing the indicator variable z we have:

$$p(x, c, z|y, \Theta, \Lambda) = \prod_{k=1}^{K} (\alpha_k P(c|\theta_k)\phi_k(x|y))^{z^k} \tag{4.12}$$

and we see that (4.12) is very similar to (4.5) except that the likelihood is conditionally dependent on the input y.

However, each expert filter requires a prior information over x_n ($x_{n-1|n-1}$), and we can not straightforwardly exploit the EM training proposed in (Xu et al., 1995). Three possible methods exist for computing the prior distribution: at first, the system providing an accurate measurement of the state can be directly used as a prior with low noise. Alternatively, we can decide to run inference on each filter independently, each of which reusing its previous output. And finally, we can run inference on the whole model, and compute before each iteration of the EM algorithm the gating weights $\frac{\alpha_k P(c|\theta_k)}{\sum_{j=1}^{K} \alpha_j P(c|\theta_j)}$, and the resulting mixture state belief.

In practice, this last solution provides the best results, and is consequently adopted. The EM based training approach proposed in (Xu et al., 1995) is then exploited, at the exception that the maximization step then only consists in minimizing the first term of result (4.8). Setting partial derivatives w.r.t to α_k, m_k, Σ_k to zero, and using Lagrangian multiplier to introduce the constraint $\sum_k \alpha_k = 1$, we obtain new estimates (Ramamurti and Ghosh, 1998):

$$\alpha_k = \frac{1}{N} \sum_n h_k(x_n, y_n) \tag{4.13}$$

$$m_k = \frac{\sum_n h_k(x_n, y_n) y_n}{\sum_n h_k(x_n, y_n)} \tag{4.14}$$

$$\Sigma_k = \frac{1}{d} \frac{\sum_n h_k(x_n, y_n) \| \ y_n - m_k \ \|^2}{\sum_n h(x_n, y_n)} \tag{4.15}$$

Using these new parameters, we then alternatively repeat inference and an iteration of EM until convergence, *i.e.* when changes in the parameter values become insignificant.

Generative training

In generative training, we focus on modeling the observation distribution $p(y_n|x_n)$. That is, each expert corresponds directly to a mapping from x_n to a subset of measurements within y_n. In this context, we should inform the reader that consequently, the name of the input and output variables of the

models are inverted relatively to the basic mixture of expert model depicted previously. However, training the gating network can be straightforwardly done with the same EM algorithm (Xu et al., 1995).

The main difference in the generative approach is that mixing is applied to different observation functions. As will be discussed more precisely in chapter 6, the generative representation yields more complex inference as there exists an active trail $x_t \rightarrow y_t \leftarrow k$. Thus, an exact Bayesian treatment does not allow for the straightforward exploitation of the recursive filtering equations. As discussed in chapter 3, this clearly shows that discriminative training also influences the manner we run inference in the model. Here, we shall note that it simplifies the exploitation of the model as it allows us to use the standard filtering equations.

4.3.4 Exploiting non-linear filters

Both the learning and inference methods discussed in sections 4.3 and 4.2.2 exploit the availability of a closed-form expression for the filters output distribution. Especially, fast and efficient mixing of the expert filters output is done thanks to the GPB approximation for Gaussian distributions. If non-linear filters have to be used, the proposed learning and inference methods can be straightforwardly exploited, for example by exploiting the UKF algorithm that still provides (approximate) Gaussian outputs. Note however that the guarantees concerning the error propagation due to the GPB approximation provided in (Boyen and Koller, 1998) is, to our knowledge, not valid if the output distributions of each filter is already an approximation.

For non-linear and non-Gaussian filters requiring stochastic approximation, propagation of the particles is a more complex problem that has not been studied yet. Mixing the filters output and training of the gating network require new adapted methods, that are kept for future work. Furthermore, until now we considered that the state estimate can be depicted by an unimodal distribution. Some applications, for instance multi-target tracking, however require to maintain a multimodal representation of the state, and this specific case is also kept for future work.

4.4 Experiments

In this section we aim at demonstrating the feasibility of the proposed approach on simulated and real data in the context of altitude estimation of a UAV. These experiments also aim at showing that the joint set of sensor measurements used for navigation augmented with relevant contextual information (i.e. other sensor measurements, or other internal data), provides a succinct yet rich representation of the perception context that can be used as the input of our context-dependent selection model. Intuitively, we can

see this joint set of measurement values as the minimum information relating the influence of the current context over sensor performance. In other words, the filter observation input signal already contains useful clues about the context.

4.4.1 Simulation

We first illustrate the system ability to learn decision rules according to sensors characteristics. This simple example reproduces the take-off and landing phases of a vertical takeoff and landing UAV. Three sensors provide direct measures of the altitude with different characteristics, such as observation noise, outliers occurrences and measurement range thresholds (Fig. 4.7).

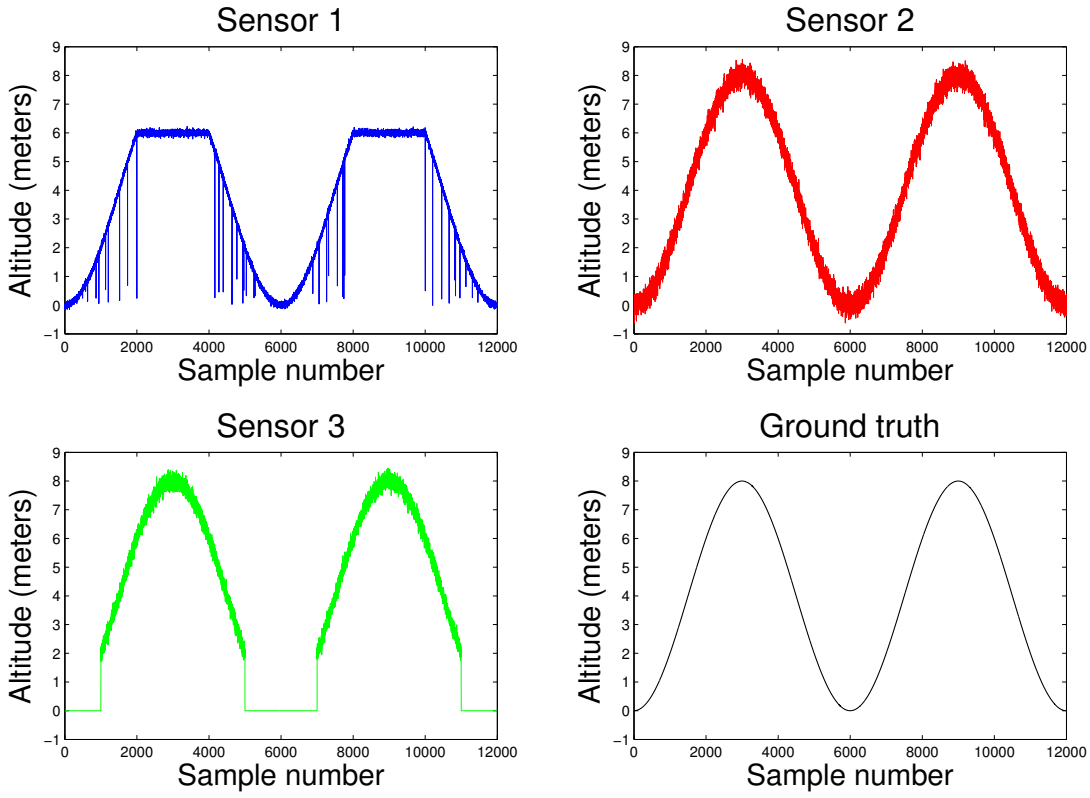


Figure 4.7: Altitude measurement provided by three simulated sensors. Sensor 1 reproduces typical ultrasonic measures, low observation noise, strong outliers occurrences and maximum range threshold. Sensor 2 permanently provides measures with high observation noise. Sensor 3 does not provide any relevant measure before reaching 2 meters.

All filters individually exploit a distinct sensor, and share a common constant velocity transition model. We train the gating network on a dataset of 12000 samples reproducing two subsequent take-off/landing sequences. The EM algorithm takes 50 iterations to converge with a convergence threshold of 10^{-5}.

The final estimate and associated uncertainty boundaries for the validation set is shown in Fig. 4.8. As we can see, the gating network learned to switch between sensors in order to reject outliers and to take into consideration each sensor measurement range. As expected from the mutual competition between experts introduced during the learning step, the gating network tends to assign binary weights (cf. Fig. 4.9). Hence mixing only operates during transition phases. As a consequence, the system output provides consistent estimation but does not benefit from estimation uncertainty reduction that could be provided by direct measure fusion.

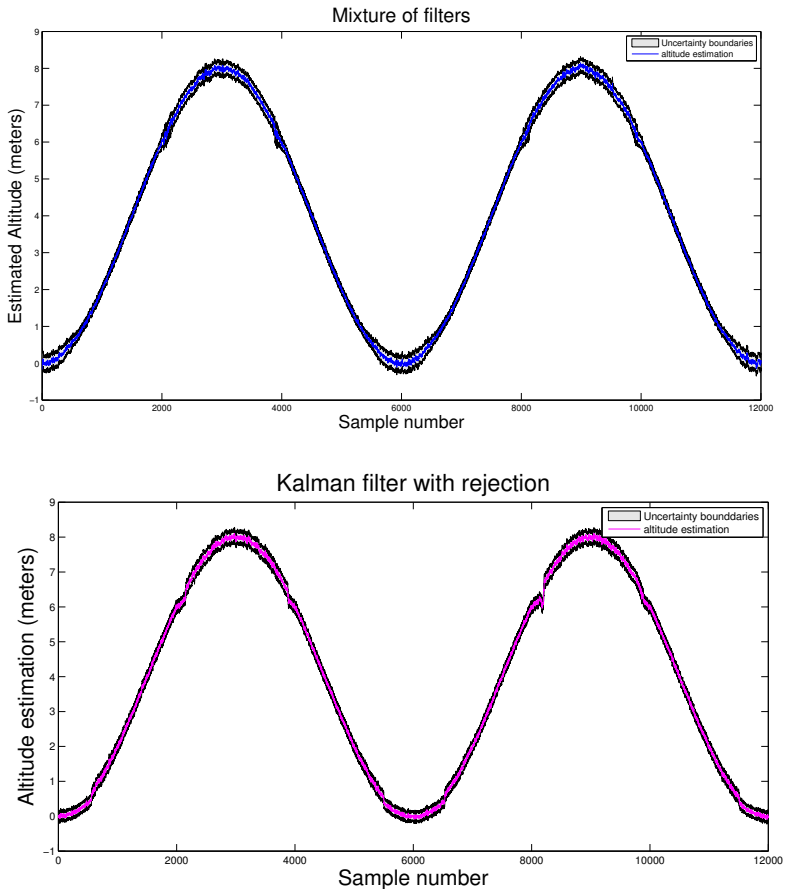


Figure 4.8: Estimated altitude on validation dataset for the mixture of filters and the Kalman filter with outlier rejection.

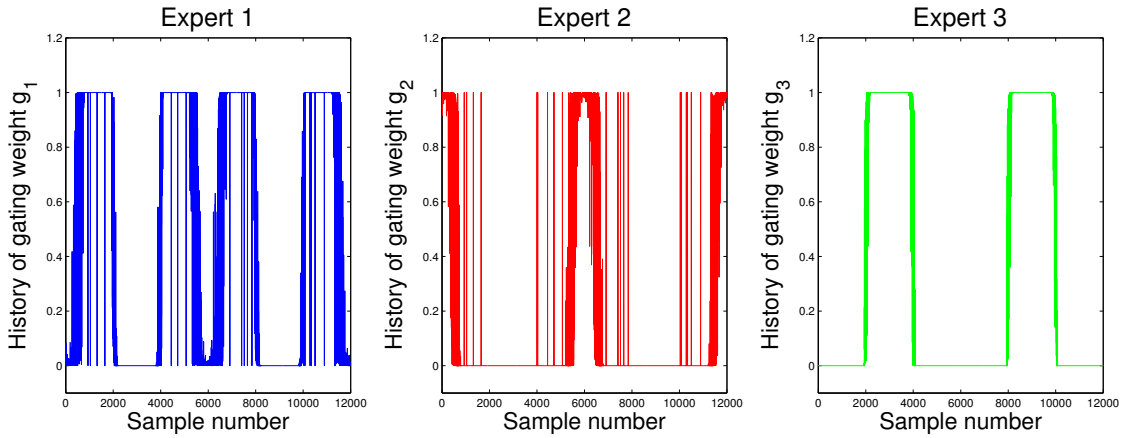


Figure 4.9: Gating weights history on validation dataset. Each expert k corresponds to the exploitation of the associated sensor k in Fig. 4.7.

We compared the ME approach with a classical Kalman filter enhanced with 3-sigma rejection on all sensors. As shown in Fig. 4.8, this approach can provide similar results with appropriately tuned filter parameters. However, the efficiency of such methods proves to be unsound, especially as small changes in filter parameters or rejection threshold can lead to strong divergence of the estimation output. Meanwhile we observed that even if changes in filter parameters could significantly modify localization of kernels in the input space, the ME approach invariably provides consistent output thanks to its adaptation capability.

4.4.2 Real data

We now use datasets acquired on a paparazzi quadrotor UAV (Brisset et al., 2006). Datasets consist of $50Hz$ synchronized altitude measures provided by an ultrasonic sensor and a barometer as well as accelerations on 3 axis provided by the embedded IMU. Altitude truth is given by a motion capture system. As we can see in Fig. 4.10, the ultrasonic sensor presents strong and frequent outliers we know to be related to thrust level. We also suppose that the barometer offset is known.

Without additional understanding of the perturbations generated on ultrasonic sensor measures, we apply the mixture of experts framework to show its ability to learn to filter these outliers, and improve estimation accuracy. For this application we use 3 different experts: one expert based on ultrasonic measures, an other based on barometer measures, and a last one based on both ultrasonic and barometer measures. As we know the presence of outliers in ultrasonic observations is correlated to the thrust, we provide 3 inputs to the gating network: both sensor measures and the thrust command. We compare this method to a Kalman filter using 3-sigma

rejection scheme on ultrasonic and barometer measures. All these filters share the same constant velocity transition model and observation noise. We train the system on a dataset of 5000 samples. After 50 iterations the EM algorithm reaches the convergence threshold fixed to 10^{-6}. Note that for both simulated or real data experiments, this corresponds to an average convergence time of 250 seconds.

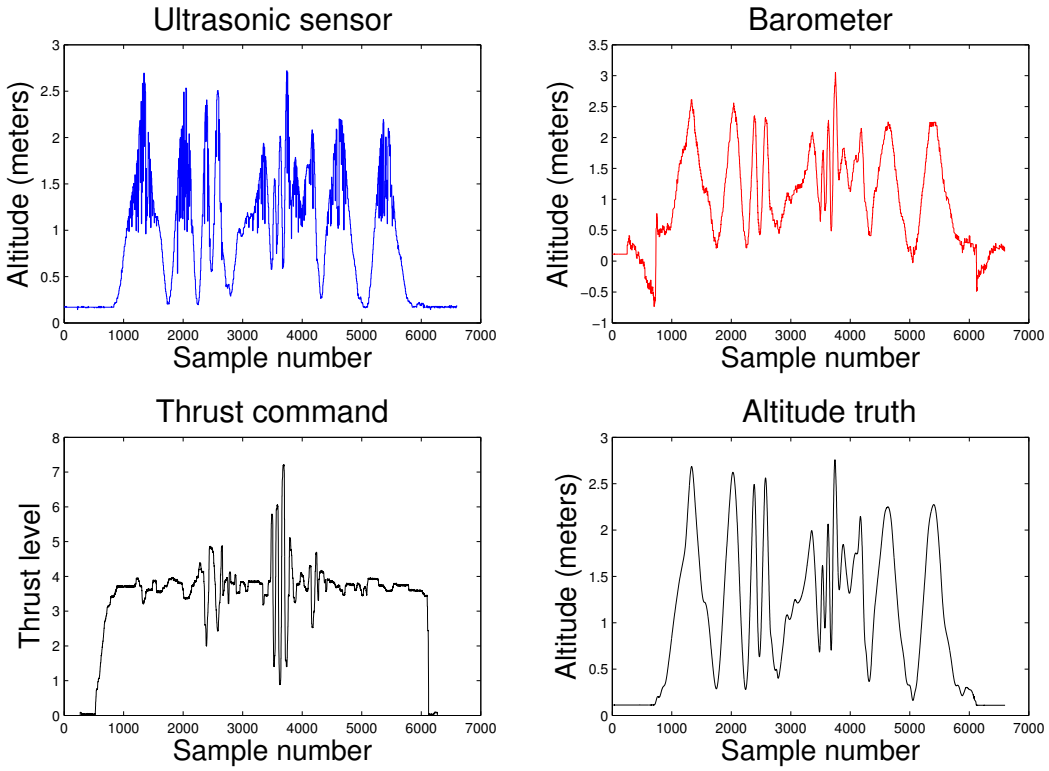


Figure 4.10: Gating network inputs and altitude truth for validation dataset.

The experiments show that the learned parameters generalize well on different validation sets, always providing similar performances. As we can see on Fig. 4.11, some outliers are not perfectly filtered. These outliers are presumably localized in unexplored regions of the input space, implying that the rejection capability could be improved by using a larger training set. On the validation set corresponding to Fig. 4.10, the system provides the best RMS estimation error with a value of 0.135. The filter with rejection provides an RMS error of 0.207. If only the sensor measurements are used as input for the gating network, we obtain an RMS error of 0.150. This result confirms that there exists a causal relationship between the thrust

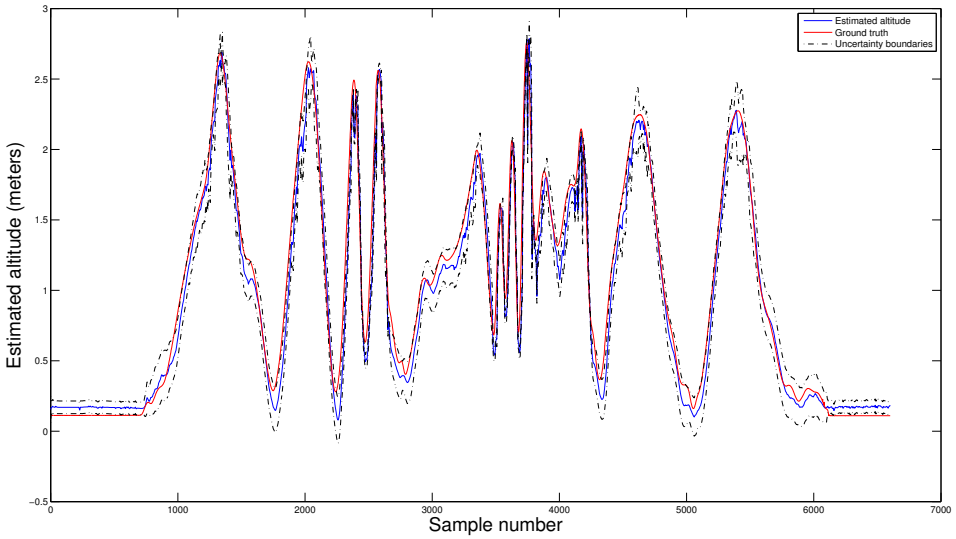


Figure 4.11: Altitude estimation and uncertainty boundaries using mixture of Kalman filters on validation set.

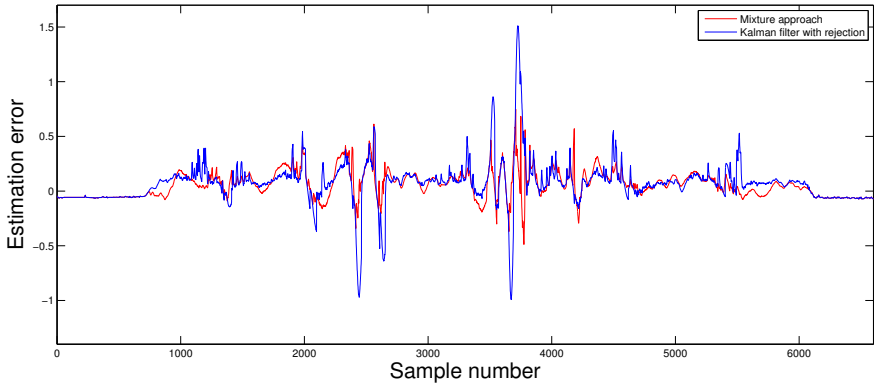


Figure 4.12: Estimation error relative to altitude truth for mixture of Kalman filter (in red) and Kalman filter with 3-sigma rejection (in blue).

command and the utility of ultrasonic measures, and illustrates the ability of the suggested approach to take it into account as well.

The estimation error improvement provided by the mixture approach (shown in Fig. 4.12) can be explained by the sensor selection process. For example our model learned to assign more weight to the ultrasonic sensor as the UAV gets closer to the ground, and usually promotes the barometer for higher altitudes, where outliers on ultrasonic measures are more likely

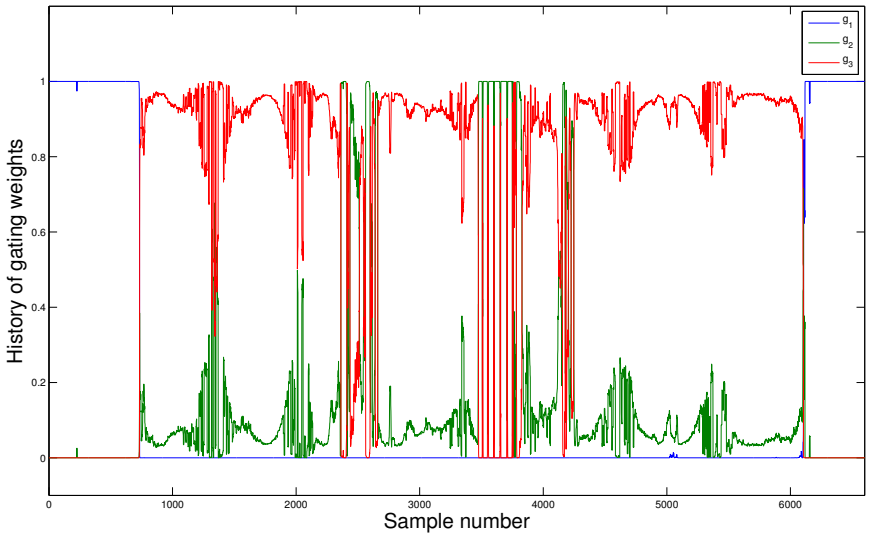


Figure 4.13: Weights history for expert 1 (ultrasonic sensor + barometer), expert 2 (ultrasonic sensor) and expert 3 (barometer).

to appear. We also notice that, due to its estimation latency, the barometer measures are more relevant for small velocities. This is why, based on the strong thrust command value, our approach reduces estimation error by selecting the ultrasonic sensor during the fast transition phase between sample 3000 and 4000. The error difference provided by the Kalman filter here again results from its sensitivity on filter parameters. The noise term on the transition model should reflects the dynamics of the UAV, but as rejection is based on the innovation term, high noise terms reduce rejection capability on small outliers. At the same time, low values can conduct to correct measurement rejection during high dynamic maneuvers. This explains the highest error peaks on Fig. 4.12 where the estimation latency introduced by the use of the barometer punctually becomes coherent with ultrasonic outliers, and make the filter diverge until the altitude decreases. In this more complex example, all the sensor specificities can not be handled by an appropriate parameter tuning, and more subtle decisions rules as encoded in the gating network prove to be more suitable.

4.5 Conclusion and remarks

4.5.1 Summary

We demonstrated that the mixture of expert framework can be applied to the measurement selection problem. In this approach, the gating network dis-

covers the different operating contexts and encodes knowledge about sensor reliability through the gating probabilities distributions parameters. This feature enables the system to automatically select the best suited estimation output, improving robustness regarding filter parameters inaccuracies and inherent sensor characteristics.

In practice, the model robustness is mainly due to the goal-oriented training step *i.e* optimization is done with respect to the output estimate accuracy. This is because optimization of the gating network parameters consists in finding the best consensus between the activation regions so as to ensure the most accurate output. It can be easily extended to more complex configurations by increasing the gating network input space.

Simultaneously, we demonstrated that the raw measurements within the observation variable of the model can provide an indirect but sufficient representation of the context influence. In practice, this means that the model learned to recognize distinctive subset of patterns within the observation vector that correspond to modification of the measurements reliability. In more complex cases, *i.e* when a common measurement pattern corresponds to different contexts, and thus different gating weights, it is necessary to introduce additional inputs to the gating network.

4.5.2 Further issues

Input synchronisation

In its original implementation the ME framework inputs are synchronized, and the gating network bases its decision on a joint set of observations. For experiments, we simulated synchronous observations by forcing sensors to provide measures at a defined frequency. As seen in the previous section, this approach doesn't affect the framework ability to make decisions, mostly because the gap between the input frequencies stays low. However, if the difference in frequencies become too important, the system wouldn't provide relevant decisional capabilities.

Complexity of the activation regions boundaries

An other constraint, directly imposed by the Gaussian kernel model, is the unimodal distribution of the activation regions. As seen earlier, one strength of the approach is that the training step automatically defines the filter expertise region. From the robustness perspective, this is a substantial benefit. However, this does not ensure that we always provide the optimal[1] output, since a trade-off has to be made between the experts, meaning that a filter may be underexploited in some areas of the input space. This phenomenon

[1]Where optimality consists here means providing the best estimate given the available measurements

is also fostered by the indicator variable z, promoting mutually exclusive activation regions, and thus decreasing the mixing capability of the model. As observed during the experiments, the gating weights behave more like binary activators than real mixing coefficients. Under specific configurations, this means that the model will not provide the optimal estimate. This would require to model the gating probabilities with more complex models, like the HME, Gaussian mixture models (Yuan and Neubauer, 2009) or more preferably nonparametric models like Gaussian processes (Yuksel et al., 2012). For the altitude estimation test case, we however noticed that the localized Gaussian kernels provide satisfying results.

Training issues

The EM algorithm provides fast and guaranteed convergence to a local maximum of the log-likelihood. In our application, the locally optimal parameters we found invariably provided relevant gating capabilities. However, in different applications involving more expert filters and a context input space of higher dimensionality, it may be necessary to improve the basic EM algorithm and avoid non-global optima. An interesting solution (among others) is proposed in (Ueda et al., 2000) with the *split and merge EM* algorithm (SMEM). Roughly, this algorithm adds an additional step within the EM iterations which consists in moving some experts from regions in the input space in which there are too many activation regions to regions in which there are too few. This provide more successful results than methods which proceeds by continuous moves because the likelihood is locally higher within intermediate locations, acting as a attractor.

4.5.3 Future directions: Towards non-parametric decision boundaries

The proposed approach is instructive in that it allowed us to confirm that context-dependent measurement selection is a relevant approach. However, the conclusions made after exploiting the localized mixture model naturally lead to an extension of the modeling expressiveness of the gating network for splitting the input space more precisely. As seen earlier, two methods can be exploited within this objective: the multi-layered HME models, or directly introducing more complex activation region models.

Following this idea, we now decide to explore an alternative approach, in which we simply map the sensor selection task to a classification problem. This allows to readily exploit some state of the art classification models, among which we focus on nonparametric and sparse models. This approach is discussed in the next chapter.

Chapter 5

Context-dependent measurement selection: An approach based on classification

Contents

This chapter presents an alternative solution to the context-dependent observation selection task. This method shares many characteristics with the model suggested in chapter 4. The main objective is to encode knowledge about the utility of each element among a bank of models in order to select the best measurement subset at runtime. Here, precise decision boundaries are defined in the context input space with the Relevance Vector Machine which efficiently provides good balance between model complexity and computational efficiency. This chapter consequently starts with a sound introduction to the Relevance Vector Machine which will also be exploited in the next chapter. This chapter presents in details the approach proposed in (Ravet et al., 2014).

5.1 A new approach based on classification

In chapter 4, measurement selection was achieved through the exploitation of the Mixture of Experts model. This framework provides an elegant solution to the measurement selection problem as it models the existence of different contexts through an additional hidden variable. During the learning phase, this variable fosters competition among filters, and results in a consensus in the distribution of the activation regions that usually provides great robustness. If the model we use to define the activation regions is too simple, this consensus may however result in a suboptimal exploitation of the experts. For instance, this can happen if the optimal activation region of a filter is discontinuous and we still exploit the unimodal Gaussian kernels in the gating network. Thus, we have no guarantee that the measurements are exploited in any region of the input space where they are providing the optimal output.

Alternatively, we can see the measurement selection task as a simple classification problem: we want to learn a mapping from the context variable input space (continuous) to the best measurement subset (discrete). Here, the interpretation of the 'best' measurement subset depends, of course, on the criterion we want to optimize. We consider two different criteria which, once again, correspond to optimizing the resulting state-observation model generatively, or discriminatively. In the generative case, this corresponds to optimizing the classifier parameters so as to obtain the best observation likelihood $p(y_t|x_t)$ over the dataset. When training aims at maximizing the conditional likelihood $p(x_t|y_1, \ldots, y_t)$, we refer to this method as discriminative training. In both cases, we can straightforwardly exploit some usual classification methods by evaluating at first the desired class target through a simple exhaustive evaluation of the performance of each measurement subset. Note that in the Mixture of Experts based approach, the performance of each subset (expert) was indirectly exploited during the training step in order to infer the hidden variable z. While in this new approach, we di-

rectly use the raw performance of each measurement subset as the selection criterion.

The core problem remains in the definition of the precise activation region of each measurement subset. Multiple methods exist in order to split the input space, but the requirements of our application lead to a specific family of methods. At first, we have no prior information about the functional form underlying the activation region boundaries. This is because the nature of the context is not understood, and neither is its impact on the measurement reliability. Thus, exploiting parametric models might be very limiting in some cases, even if it is a highly problem-dependent issue. Also, we aim at exploiting the complete model online, which requires the measurement selection to be done without degrading the computational efficiency relatively to that of the recursive equations.

Here, we consider non-parametric models as the best solution for defining accurate activation regions. We also believe that non-parametric models provide strong genericness and thus require no problem-specific tuning for different filter implementations. Non-parametric models exploit the concept of the *memory-based* approach, in which the prediction for a new input relies on the whole training data. Gaussian Processes (GP) (Rasmussen and Williams, 2005) are one of the most popular concept in non-parametric modeling, and have been used for learning the observation and prediction distributions of a state-observation model (Ko and Fox, 2009; Deisenroth et al., 2012). GPs however suffer from high computational complexity, which makes them irrelevant for real-time state estimation if the basic training methods are used. Their time efficiency has been increased by forcing *sparsification* of the dataset during training (Quiñonero-candela et al., 2005), but this operation remains complex and the resulting computational cost for a new prediction may remain prohibitive for online applications (Ko and Fox, 2009).

Data sparsification has become a very popular concept with the advent of the *Support Vector Machines* (Cortes and Vapnik, 1995), in which only a small subset of samples within the training set is used for new predictions (the support vectors). Within the Bayesian community, data sparsification has been addressed through a specific training method referred to as *automatic relevance determination* (ARD) or *sparse Bayesian learning*. The Relevance Vector Machine (RVM) introduced in (Tipping, 2001) exploits this sparse learning technique combined with the *kernel method* detailed hereafter. This results in a model that provides better generalization capabilities than the SVM with significantly increased sparsity. In fact, RVM can be seen as a specific instantiation of GP, where training automatically introduces sparsification. Also, the RVM brings the usual benefits of the probabilistic models and thus provides predictive distributions instead of point predictions. This means that each prediction output is jointly pro-

vided with a measure of its uncertainty that depends on both the process noise observed in the dataset, and on the correlation between the new input and the training data. Regarding this feature, we will see that the RVM has a counter-intuitive behavior when the new input is too far from the training data, so that the prediction uncertainty decreases. However, we believe that RVM is a key solution for learning non-parametric and computationally efficient models for online measurement selection. In the next section, we provide a detailed background about RVM before introducing the new measurement selection method, which was originally introduced in (Ravet et al., 2014).

5.2 Background on the Relevance Vector Machine

While we will ultimately exploit the model for classification in this chapter, it is however simpler to introduce the concepts exploited in RVM starting with the regression task. Classification can then be introduced as an extension of the regression form of the RVM. Note that regression will be exploited for the noise adaptation problem in chapter 6, thus the background given in this section is of general purpose in this manuscript.

5.2.1 Extended linear models

We consider a regression task in which we are given a training set $\mathcal{D}$ containing N observations $\{x_n, y_n\}_{n=1}^{N}$, where x is an input vector of dimension D and y denotes a scalar output. In the Bayesian approach, the regression task consists in learning a conditional distribution that explains the relationship between the inputs and the outputs. Ultimately, we are only interested in making prediction of the output y^* for a new input x^* through the exploitation of the learned conditional distribution $p(y|x)$.

Bayesian linear regression

Here, we start by depicting the classical linear regression method under the Bayesian approach. The expressiveness of this model being very limited, we see how it can be extended by re-mapping the input space on an alternative feature space. This extension leads to an equivalent representation known as the kernel substitution which plays a central role in non-parametric modeling, and is the backbone of both the SVM and the RVM.

We start with the simplest regression method that is the standard linear model with Gaussian noise:

$$y = x^{\intercal}\mathrm{w} + \epsilon$$

where w is a vector of weights and $\epsilon \sim \mathcal{N}(0, {\sigma_r}^2)$ the additive noise component. By introducing this last term, we assume that y may differ from $x^\intercal \mathrm{w}$, following an i.i.d Gaussian distribution.

Training generally consists in finding the unknown model parameters w and σ_r. In the following discussion, the noise covariance is supposed to be known and we focus exclusively on w as it is sufficient to illustrate the kernel substitution concept. If we now write $\mathbf{X}$ the $D \times n$ matrix containing all the training input vectors and $\mathbf{y} = \{y_1, \ldots, y_N\}$, we can write the model likelihood:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathrm{w}) &= \prod_{n=1}^{N} p(y_n|x_n, \mathrm{w}) \\ &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_r} \, exp\Big(- \frac{(y_n - x_n^\intercal \mathrm{w})^2}{2{\sigma_r}^2} \Big) \\ &= \frac{1}{(2\pi{\sigma_r}^2)^{N/2}} \, exp\Big(- \frac{|\mathbf{y} - \mathbf{X}^\intercal \mathrm{w}|^2}{2{\sigma_r}^2} \Big) \end{aligned}$$

Thus the likelihood is a N-dimensional multivariate Gaussian with mean given by $\mathbf{X}^\intercal \mathrm{w}$ and covariance given by $\sigma_r I$ (where I is the $N \times N$ identity matrix). In the Bayesian approach, we treat the unknown parameters as hidden random variables, and introduce a prior distribution over w. A common prior is a Gaussian centred in zero with covariance Σ_r so that $p(\mathrm{w}) \sim \mathcal{N}(0, \Sigma_r)$. This specific distribution may seem arbitrary, but it actually brings some interesting properties that will be detailed later. Learning then consists in finding the parameters in Σ_r which maximize the dataset likelihood. Once we learned the parameters, we can compute the posterior distribution over w and use it for prediction.

To compute the posterior distribution over w, we first note that $p(\mathbf{y}|\mathbf{X}, \mathrm{w})$ is Gaussian and linear in w, where the prior distribution over w is also Gaussian. Thus we can exploit once again the linear Gaussian model properties A.2 to evaluate the posterior:

$$p(\mathrm{w}|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}(m_w, \Sigma_w)$$

where

$$\begin{aligned} m_w &= {\sigma_r}^{-2} \Sigma_w^{-1} \mathbf{X}\mathbf{y} \\ \Sigma_w &= {\sigma_r}^{-2} \mathbf{X}\mathbf{X}^\intercal + \Sigma_r^{-1} \end{aligned}$$

From then on, we can make predictions for new inputs by marginalizing out w in $p(y^*|x^*, \mathrm{w})$ with respect to its posterior:

$$p(y^*|x^*, \mathbf{X}, \mathbf{y}) = \int p(y^*|x^*, \mathrm{w}) \, p(\mathrm{w}|\mathbf{X}, \mathbf{y}) \, \mathrm{dw}$$

where we notice that the two components within the integral are Gaussian. Thus we can straightforwardly compute the marginal distribution $p(y^*|x^*, \mathbf{X}, \mathbf{y})$ using A.1:

$$p(y^*|x^*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(m^*, \Sigma^*)$$

where

$$m^* = \frac{1}{{\sigma_r}^2} x^{*\intercal} \Sigma_w^{-1} \mathbf{X}\mathbf{y}$$

$$\Sigma^* = {\sigma_r}^2 + x^{*\intercal} \Sigma_w^{-1} x^*$$

The predictive distribution is thus an other Gaussian whose mean is logically given by the product of the new input by the posterior mean of the weight. The first term in the prediction uncertainty corresponds to the noise in the data, and the second term is a quadratic form of the new input with the posterior covariance of the weights. Thus, the prediction uncertainty is proportional to the noise in the data and to the uncertainty associated with the weights. Note that the last term also logically increases with the magnitude of the input.

Extended linear model and the kernel method

Clearly, the basic Bayesian linear model has significant limitations in terms of modeling capabilities. It however provides interesting analytical properties that have been exploited in the development of more complex models. For instance, one very simple idea for extending the model expressiveness consists in projecting the input into a new high dimensional space before applying the linear model. This is usually done by defining a nonlinear function ϕ that maps the D dimensional input vector x into a new M dimensional *feature space* where $M > D$. The regression model is now written:

$$y = \phi(x)^\intercal \mathrm{w} + \epsilon \tag{5.1}$$

where the vector of weights w is now M-dimensional. Interestingly, the Bayesian analysis for this new model is identical to the basic linear model, and if we denote the aggregation of columns vectors $\phi(x)$ for all x in $\mathcal{D}$ by Φ, we can show that the prediction distribution can now be written (Rasmussen and Williams, 2005)

$$p(y^*|x^*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(m^*_\phi, \Sigma^*_\phi)$$

where

$$m^*_\phi = \phi(x^*)^\intercal \Sigma_r \Phi (K + \sigma_r^2 I)^{-1} \mathbf{y}$$

$$\Sigma^*_\phi = \phi(x^*)^\intercal \Sigma_r \phi(x^*) - \phi(x^*)^\intercal \Sigma_r \Phi (K + \sigma_r^2 I)^{-1} \Phi^\intercal \Sigma_r \phi(x^*)$$

where we defined $K = \Phi^\intercal \Sigma_r \Phi$.

As can be seen, the feature space into which we projected the input is always exploited in the form of a product $\phi(x)^\intercal \Sigma_r \phi(x')$ where x and x' may correspond to points within the training set, or to the new test point. In the dual kernel representation, we define a kernel[1] function so that $k(x, x') = \phi(x)^\intercal \Sigma_r \phi(x')$. The kernel function then directly represents the result of the inner product of $\phi(x)$ and $\phi(x')$ with respect to Σ_r. If we properly chose the kernel function, we can then substitute the potentially complex inner product in the feature space by the evaluation of the equivalent kernel. This general idea of processing the input data directly through the evaluation of a kernel function is sometimes referred to as the kernel trick.

Multiple kernels have been developed within the different applications of kernel regression, among which the Gaussian kernel which takes the form:

$$k(x, x') = exp\Big(-\frac{1}{2}(x - x')^\intercal \Sigma^{-1}(x - x') \Big)$$

Under its exact form, the Gaussian kernel requires the full parametrization of the covariance Σ, which is a complex task. In practice, it is common to assume that Σ is diagonal, so that we have:

$$k(x, x') = exp\Big(-\frac{1}{2}\sum_{i=1}^{D} \frac{1}{\sigma_j}(x_j - x'_j)^2 \Big) \tag{5.2}$$

where σ_j is referred to as the length scale on dimension j. Finally, when σ_j is considered to be identical on all dimensions, we obtain the isotropic kernel:

$$k(x, x') = exp\Big(-\frac{\| x - x' \|^2}{2\sigma^2} \Big) \tag{5.3}$$

which is known as the *radial basis function* (RBF) kernel. Note that for simplicity, this last kernel is often preferred over more complex parametrizations.

Practically, the σ_i parameters can be determined by manual tuning. More precisely, it is commonly done by assessing the performance of the model for distinct values of the σ_i's through $cross - validation$[2] and subsequently choosing the values that provided the best results. Alternatively, the length scale parameters can also be optimized during the training step. This latter approach especially makes sense when we prefer to use individual parameters σ_i instead of an isotropic kernel since manual tuning may quickly become inapplicable for input spaces of high dimensionality.

[1] Also called covariance function

[2] Cross validation is a simple technique where we divide the training set into N groups. The model is trained N times on a training set containing $N - 1$ groups before assessing its performance on the remaining N^{th} group. The performance of the model for the N runs is then averaged.

We can provide two basic interpretations that justify the exploitation of Gaussian kernels in extended linear models. In a more intuitive analysis, we see that the Gaussian kernel provides a measure of the distance between two input points. In the case where we have no clear idea about a relevant feature space for projecting the input data, an alternative approach is to exploit the correspondence between input objects. For example if we are not able to classify an object through a set of specific characteristics (features), we can however evaluate how similar it is to another, and thus rely on this score to make a decision. Consequently, the Gaussian kernel can be seen as a similarity score between two inputs, this score being exploited to make prediction about new inputs by comparison to the samples seen earlier during the training procedure.

More formally, it can be proved, using the Mercer's theorem (Rasmussen and Williams, 2005), that working with the Gaussian kernel is equivalent to projecting the input variable on an infinite feature space. Note that in this case the weight vector w is consequently infinite dimensional, and this level of complexity could not be handled without kernel substitution. The resulting regression model is then very powerful and generic, making the Gaussian kernel one of the most popular choice in kernel methods. Consequently, we also adopt the Gaussian kernel in our first investigations concerning the measurement selection task. However, the RVM model itself is generic, and does not impose the exploitation of this specific kernel. For more details about the different kernel functions and the kernel substitution, an extensive discussion is provided in (Schölkopf and Smola, 2002).

5.2.2 RVM for regression

The Relevance Vector Machine, as described in (Tipping, 2001), is an extended linear model of the form (5.1). As a kernel machine, the likelihood of the complete training set can be written using the dual kernel representation so that we have

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma_r) = \frac{1}{(2\pi {\sigma_r}^2)^{N/2}} \, exp\Big(- \frac{1}{2\sigma_r^2} \parallel \mathbf{y} - \mathbf{\Phi w} \parallel^2 \Big) \tag{5.4}$$

with $\mathbf{\Phi}$ the design matrix whose elements are given by the kernel function so that $\mathbf{\Phi} = [\phi(x_1), \ldots, \phi(x_N)]^\intercal$ with $\phi(x_i) = [1, k(x_i, x_1), k(x_i, x_2), ..., k(x_i, x_N)]^\intercal$.

The RVM however differs from the classical linear model because individual Gaussian priors are defined over each element w_i of $\mathbf{w}$:

$$p(w_i|\alpha_i) \sim \mathcal{N}(0, \alpha_i^{-1}) \tag{5.5}$$

where, following the notation in (Tipping, 2001), α_i is the precision[3] hyperparameter corresponding to the weight element w_i. This yields an indepen-

[3]Recalling that the precision is the inverse of the variance

dent joint prior distribution over $\mathbf{w} = (w_1 ... w_{N+1})^\intercal$:

$$p(\mathbf{w}|\alpha) \sim \mathcal{N}(0, A)$$

where $A = diag(\alpha_1, \ldots, \alpha_{N+1})$. The resulting graphical model is depicted in the figure hereafter.

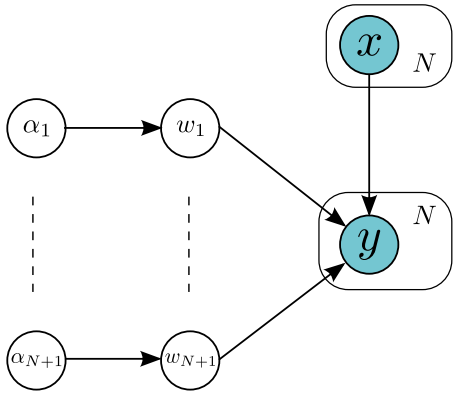


Figure 5.1: Graphical representation of the RVM regression model.

As for the linear model, the posterior distribution of $\mathbf{w}$ can be evaluated using A.2 and takes the form:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha, \sigma_r) \sim \mathcal{N}(m_w, \Sigma_w) \tag{5.6}$$

where

$$m_w = {\sigma_r}^{-2} \Sigma_w^{-1} \Phi^\intercal \mathbf{y}$$

$$\Sigma_w = (A + {\sigma_r}^{-2} \Phi^\intercal \Phi)^{-1}$$

Sparse Learning

It is possible to define proper conjugate priors for the α_i's and σ_r, *i.e* a Gamma prior for the α_i's and an inverse Gamma prior for σ_r. However we follow the idea suggested in (Tipping, 2001) and take non-informative uniform priors that correspond to the limit case where all the parameters of the Gamma prior are set to zero. Note that the convenient prediction equations described for the linear model could be obtained because we had fixed values for the parameters Σ_r and σ_r. Here, the posterior distribution over the α_i's and σ_r is consequently approximated by a point estimate (corresponding to delta functions at the mode of the posterior distribution). Thus, if we seek for the maximum of the posterior distribution of the unknown parameters, we want to find the points α_{max} and $\sigma_{r\,max}$ that maximize the posterior (using the Bayes rule):

$$p(\alpha, \sigma_r|\mathbf{y}, \mathbf{x}) \propto p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_\mathbf{r}) p(\alpha, \sigma_r)$$

where, $\alpha = \{\alpha_1, \ldots, \alpha_{N+1}\}$.

Since we chose an uniform prior distribution $p(\alpha, \sigma_r)$, finding the maximum of the posterior distribution is then equivalent to maximizing the conditional likelihood marginalized with respect to $\mathbf{w}$, *i.e* $p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_\mathbf{r})$. Representing the hyperparameters by point estimates obtained by maximizing the marginal likelihood is a method known as the *Type-2 Maximum Likelihood*, or *evidence approximation* (Mackay, 1992). In our case, the marginal likelihood can be evaluated by writing:

$$p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_r) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma_r) p(\mathbf{w}|\alpha)\, \mathrm{d}\mathbf{w} \tag{5.7}$$

which results in a new Gaussian distribution. Indeed, using A.1 we have

$$p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_r) \sim \mathcal{N}(O, C)$$

where

$$C = \sigma_r^2 I + \Phi A \Phi^\intercal$$

Training is done by optimising the usual log likelihood with respect to α and σ_r. This corresponds to optimising:

$$\begin{aligned} ln\big(p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_r)\big) &= ln\big(\mathcal{N}(O, C)\big) \\ &= -\frac{1}{2} N ln(2\pi) + ln(|C|) + \mathbf{y}^\intercal C^- 1 \mathbf{y} \end{aligned} \tag{5.8}$$

There is however no direct closed form for α_{max} and $\sigma_{r\,max}$, and consequently optimization is done through a two-step iterative procedure described hereafter. After choosing initial values for α and σ, the first step consists in maximizing the current marginal likelihood by setting the corresponding derivatives to zero. This provides new estimates α^{new} and $\sigma_r^{2^{new}}$ that are used for updating the posterior distribution of the weights. Following (Mackay, 1992), the new estimates are provided by:

$$\begin{aligned} \alpha^{new} &= \frac{1 - \alpha_i \, \Sigma_{wii}}{m_{w_i}^2} \\ \sigma_r^{2^{new}} &= \frac{\| \, \mathbf{y} - \Phi m_w \, \|^2}{N - \sum_i (1 - \alpha_i \, \Sigma_{wii})} \end{aligned} \tag{5.9}$$

where m_{wi} is the i^{th} component of the mean of the posterior over $\mathbf{w}$, and Σ_{wii} the i^{th} diagonal component of the covariance of the posterior over $\mathbf{w}$.

In the second step, we evaluate the new marginal likelihood

$$p(\mathbf{y}|\mathbf{x}, \alpha^{new}, \sigma_r^{new}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma_r^{new}) p(\mathbf{w}|\alpha^{new})\, \mathrm{d}\mathbf{w}$$

and re-estimate the new parameters. These two steps are then repeated until a convergence criterion is reached.

In practice, most of the α_i tend to large values (numerically equivalent to infinite) during the training step. Given the Gaussian distribution defined over $\mathbf{w}$ (5.5), this means that the corresponding w_i is zero mean with very low uncertainty. The corresponding component of the weight vector can therefore be pruned from the model as it is not required for making new predictions. The input values x_n associated with the remaining weights are consequently referred to as the *relevance vectors*, as they are automatically identified by the system as the points of input space being relevant for making predictions.

Alternatively, the EM algorithm can be used by considering $\mathbf{w}$ as the hidden variable of the model. This however leads to updates equations for α^{new} and σ_r^{2new} that are less efficient than the one we described when using type-2 maximum likelihood. EM thus provides significantly slower convergence, and as for any memory-based approach, this problem is worsened when the size of the training set increases. Consequently, it is customary to rely on type-2 maximum likelihood, even if we shall recall that it is an approximation of the pure Bayesian approach, since we do not optimize the exact likelihood of the model.

The mechanism of sparsification is a direct consequence of the exploitation of peculiar priors that enforce convergence of some weight components to zero, and is known as *automatic relevance determination*. Note that ARD is not specific to the RVM and can be exploited for any extended linear model. Exploitation of priors that are purposely designed for fostering sparsity is strongly debatable, as well as pruning parts of the training set. This is because an exact Bayesian treatment would still assign a distribution to the 'irrelevant' points within the training set. Also, it is not consistent with the theoretical role of the prior, which is to represent the prior knowledge we have over the variable $\mathbf{w}$. However, models like the RVM or the SVM are now widely used in many applications, where their sparsification capabilities are particularly appreciated. For more fairness regarding the probabilistic Bayesian framework, we should preferably refer to the RVM as a *semi-Bayesian* extended regression model.

While it is intuitive to understand that some weight parameters are 'attracted' to zero during the optimization (due to the specific zero mean Gaussian prior), it is not straightforward to understand why some of the hyperparameters α_i go to infinity. In (Tipping, 2001) the author provides an intuitive interpretation explaining why some components ϕ_i of an extended linear model may be pruned during optimization of the likelihood. Later, a sound mathematical analysis was provided in (Faul and Tipping, 2001), which we briefly describe in appendix C. Note that an alternative analysis,

based on the variational framework, was also provided in (Wipf et al., 2003)[4].

Extension of the training method

As described in (Tipping, 2001), the input length scale parameter of the kernel function strongly affects the performance of the model, especially for the regression task. From now on, and following the usual notation, we denote the input length scale parameters vector by η so that the isotropic Gaussian kernel (1-dimensional length scale vector) is now written:

$$k(x, x') = exp\Big(-\eta \parallel x - x' \parallel^2 \Big)$$

It is very common to adjust this parameter by hand, and only train the model with respect to the weight hyperparameters α and the noise σ_r. However, the kernel input length scale parameter can also be estimated along with other parameters during the training step. Note that for kernels of the form (5.2), tuning the corresponding parameters η_i quickly become intractable, and learning is the only relevant method.

While all the parameters (α, σ_r, η) can be optimized through conjugate gradient ascent, the updates equations (5.9) provide the most efficient method for optimizing α and σ_r. In practice it is thus more efficient to run a few iterations of gradient ascent with respect to η after each update of α and σ_r.

Note that in our application we will exploit isotropic kernels and consequently tune the length scale parameter according to the ultimate system performance.

Prediction

After the training procedure, we can use the optimal hyperparameters α_{max} and $\sigma_{r\,max}$ to make predictions for new test inputs x^* by computing the predictive distribution

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}, \alpha_{max}, \sigma_{r\,max}) = \int p(y^*|x^*, \mathbf{w}, \sigma_{r\,max}) p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha_{max}, \sigma_{r\,max})\, \mathrm{d}\mathbf{w}$$

This gives a new Gaussian distribution:

$$p(y^*|x^*, \mathbf{x}, \mathbf{y}, \alpha_{max}, \sigma_{r\,max}) \sim \mathcal{N}(m^*, \sigma^{2^*}) \tag{5.10}$$

where

$$m^* = m_{w\,max}{}^{\intercal} \phi(x^*)$$

$$\sigma^{2^*} = \sigma_{r\,max}^2 + \phi(x^*)^{\intercal} \Sigma_{w\,max} \phi(x^*)$$

[4]Since we have not described the complex specifics of the variational framework, we prefer to rely on the work by Faul and Tipping.

with $m_{w\,max}$ and $\Sigma_{w\,max}$ the mean and covariance of the posterior distribution over $\mathbf{w}$ evaluated at α_{max} and σ_{rmax}.

Note that the function ϕ is built on the kernel function localized at the relevance vectors. Consequently, when a new test point x^* lies too far from the relevance vectors, kernel functions such as the Gaussian kernels go to zero (as a measure of distance), and the predictive uncertainty reduces to the noise of the data σ_r. Similarly, the predictive mean also converges to zero, and the resulting prediction is wrong with a very low uncertainty, as illustrated in Fig.5.2.

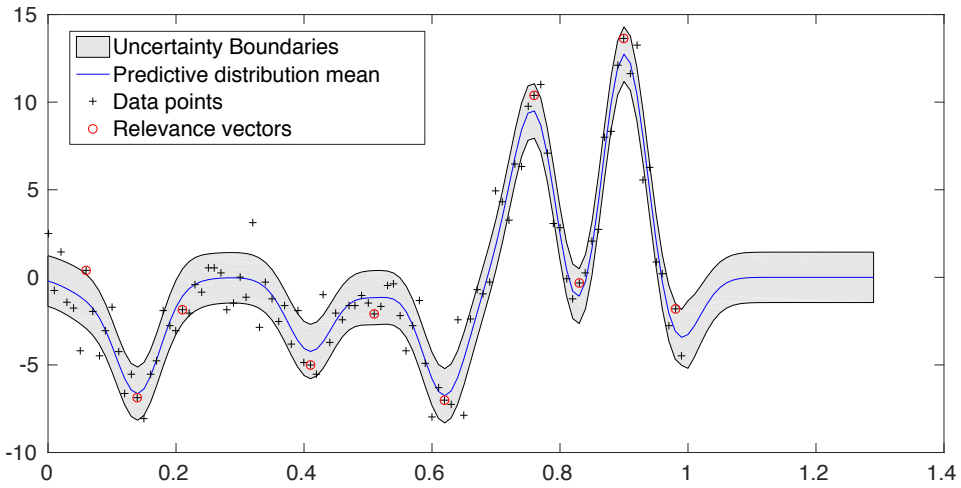


Figure 5.2: Predictive distribution of a RVM model trained on data points in $[0, 1]$. As expected the prediction mean converges to zero when extrapolating too far from the training data (here for any new input > 1) , and the variance becomes equal to $\frac{1}{\sigma_{r^2_{max}}}$.

This result is very counter-intuitive for a probabilistic model, since the user would clearly expect the output uncertainty to increase drastically when the relevance vectors do not allow for properly making any conclusion about the new input. More details about this undesirable effect can be found in (Rasmussen and Candela, 2005), where a solution for 'healing' this behavior is proposed. Roughly, this approach suggests to *augment* the basic RVM model by introducing at runtime a new component centred on the new input x^* in the vector ϕ (referred to as a basis function). In other words, it augments the relevance vectors extracted from the initial training set with a new element corresponding to the input x^*. This results in new equations for the prediction mean and variance that fix the defective behavior of the RVM in parts of the input space that were not explored during the initial training step. Unfortunately, the evaluation of the new mean and variance requires the full training set, and the model can not be considered to be sparse

anymore. This illustrates the fact that sparsity is obtained at the expense of accuracy, and that efforts still have to be done in order to improve the balance between these two aspects. Regarding this issue, the basic RVM chooses sparsity over accuracy, while on the contrary, models such as the GPs aim at accuracy but come at high computational costs.

5.2.3 RVM for classification

The classification task differs from regression in that the target values are now discrete labels representing the different classes. Practically, when we have K different classes, the target t_n is represented through a 1-of-K coding scheme, *i.e* t_n is a K-dimensional vector in which the class C_k is represented by setting all components to zero except for the k^{th} component. In other words, for the basic binary classification task where $t_n \in \{0, 1\}$, class C_1 is represented by $t_n = (1, 0)^\intercal$ and C_2 by $t_n = (0, 1)^\intercal$. In practice, and following the probabilistic approach, the target values can be represented by probabilities, meaning that some components in t_n are not strictly 0 or 1, but a normalized coefficient representing the posterior probability of each class.

The classification task consists in learning a mapping $x_n \mapsto t_n$, where the input x_n is in $\mathbb{R}^D$. Linear and extended linear models for regression have been generalized to classification, where the main difference in the analysis arises from the specific likelihood function adopted for the discrete labels. Usually, and following the Bayesian approach, the linear model is generalized through the introduction of an *activation* function f so that $p(C_k|x) = f(\phi(x)^\intercal w)$, where we project the input on a new feature space as explained before. A popular activation function is the *logistic sigmoid* $\sigma(x)$ defined by

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

which maps the real input x to the finite interval $[0, 1]$ as illustrated hereafter.

The RVM classification method originally introduced in Tipping (2001) is designed for binary classification. The likelihood function is inspired by the logistic regression model (Bishop, 2006) which takes the form

$$p(\mathbf{t}|\mathbf{w}, \mathbf{x}) = \prod_{n=1}^{N} \sigma(y(x_n))^{t_n} \left[1 - \sigma(y(x_n))\right]^{1-t_n} \tag{5.11}$$

where we defined $y(x_n) = \phi(x)^\intercal w$, and exploit the property $p(C_2|x) = 1 - p(C_1|x)$. By comparison with the regression task, we see that there is no noise on the data (σ_r). If we now introduce a prior distribution over the weight $p(\mathbf{w}|\alpha)$, we see that due to the sigma activation function, the marginal likelihood $p(\mathbf{t}|\mathbf{x}, \alpha)$ as well as the posterior $p(\mathbf{w}|\mathbf{t}, \alpha)$ do not allow

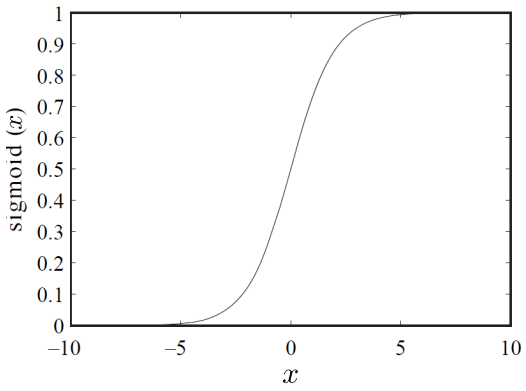


Figure 5.3: The sigmoid activation function.

for a closed-form expression anymore (recall that closed-form expression could be obtained thanks to the linear Gaussian properties (A.1) and (A.2)).

To cope with this issue, a general approach that was originally introduced in (MacKay, 1992) consists in finding a Gaussian approximation to the posterior $p(\mathbf{w}|\mathbf{t}, \alpha)$. This is done through the application of the Laplace approximation that basically fits a Gaussian distribution whose mean is centred on a *mode* of the original distribution. The Laplace approximation also provides an approximate closed-form for the marginal likelihood, whose optimization leads to updates equations for the α_i hyperparameters that are similar to (5.9). Note that the analysis of sparsity that we provided for the regression task is also applicable in this case, and consequently the training step also results in great sparsity.

In our application however, the number of classes is very likely to be higher than two, and we need to exploit multi-class methods. The original binary model can be straightforwardly extended to the multi-class setting, where the likelihood now takes the form:

$$p(\mathbf{t}|\mathbf{w}, \mathbf{x}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \sigma(y_k(x_n))^{t_{nk}} \tag{5.12}$$

We do not provide further details about this approach since we will not exploit this basic RVM model for classification. Indeed, in its original formulation, the RVM classifier scales badly with the number of classes C. More precisely, the covariance matrix of the Laplace approximation scales with C (The covariance matrix has size $MC \times MC$ with M the number of relevant vectors), and the RVM classifier is known to adapt poorly to the multi-class setting (Damoulas et al., 2008).

Note that an alternative approach to the multi-class setting consists in training multiple and concurrent binary classifiers. In the *one-versus-the-rest* approach for instance, C distinct models are trained by taking the data

corresponding to each different class as positive examples, while the data from the other $C-1$ classes are taken as negative examples. Predictions for a new input are then made by selecting the maximum output $y_c(x_n)$ among the C different classifiers. This approach suffers from multiple problems, mainly due to the fact that the different output probabilities have irrelevant scales since the models are trained independently. Also, the amount of computation required for both training and prediction is significantly higher and increases with the number of classes.

Extension for the multi-class setting

More recently, new formulations of the multi-class RVM were derived (Damoulas and Girolami, 2008; Damoulas et al., 2008) in order to improve the computational efficiency and accuracy of the original method. Note that this specific model was also developed in order to improve the modeling expressiveness through the *multi-kernel* method. This means that the original RVM formulation can be augmented with a new composite kernel $K^{\beta}(x_i, x_j)$ described as a convex combination of base kernels $K^s(x_i^s, x_j^s)$ so that $K^{\beta}(x_i, x_j) = \sum \beta_s K^s(x_i^s, x_j^s)$. This feature allows the model to handle strongly heterogeneous input data and provide very generic learning capabilities. In our application, this feature could bring significant benefits in complex settings where the context is represented through numerous and strongly heterogeneous inputs. However, during our first investigation we will focus on proving the feasibility of the classification-based approach and exploit a unique Gaussian kernel.

We now give a quick overview of the multiclass RVM (mRVM) originally proposed in (Damoulas and Girolami, 2008). This model requiring an extensive theoretical analysis and a complex background on variational methods (Girolami and Rogers, 2005), we encourage the reader to explore the work of Damoulas (Damoulas and Girolami, 2008; Damoulas et al., 2008; Damoulas and Girolami, 2009) for a deeper understanding of the approach.

The mRVM method extends the original RVM classifier through the introduction of a new continuous and C-dimensional hidden variable y that is treated as an intermediate regression target output. This method was actually introduced earlier and leads to a model known as the *multivariate probit regression* (Albert and Chib, 1993). Each component y_{cn} is here provided by a Gaussian with standard noise model so that

$$y_{cn} \sim \mathcal{N}(\phi(x_n)^{\intercal} w, 1)$$

The target variable t_n is then set according to y_{cn} through

$$t_n = i \quad \textbf{if} \quad y_{in} > y_{j_n} \quad \forall j \neq i$$

and the resulting graph is shown in Fig. 5.4.

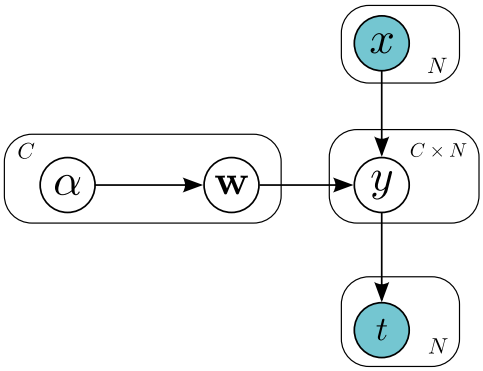


Figure 5.4: Graphical representation of the mRVM. Note that α and w are $(N+1)$-dimensional.

The specific likelihood is derived by marginalizing out the intermediate variable y (or more precisely all components y_{cn}) which gives

$$\begin{aligned} p(t_n = i|\mathbf{W}, \mathbf{x}) &= \int p(t_n = i|y_n) p(y_n|\mathbf{W}, \mathbf{x}) \, \mathrm{d}\mathbf{y_n} \\ &= \int \delta(y_{i_n} > y_{j_n} \, \forall j \neq i) \prod_{c=1}^{C} \mathcal{N}(\phi(x_n)^\intercal w_c, 1) \, \mathrm{d}\mathbf{y_n} \\ &= \mathbb{E}_{p(u)} \Big\{ \prod_{j \neq i} \Phi\big(u + \phi(x_n)^\intercal (w_i - w_j)\big) \Big\} \end{aligned}$$

where $\mathbb{E}$ represents the expectation taken with respect to $p(u) \sim \mathcal{N}(0,1)$ and Φ is known as the *probit* function, or Gaussian cumulative distribution and is given by

$$\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(\theta|0,1) \, \mathrm{d}\theta$$

The mRVM likelihood is closely related to the conventional probit regression method, except that the probit function inputs are here represented by Gaussian distributions. In other words the mRVM approach directly plugs-in the outputs of the distinct RVM regression models in the probit activation function, which in turn provides probabilistic outputs for each class membership. This probabilistic representation results in an increased complexity, as it requires to evaluate the expectation with respect to $p(u)$. Note that the evaluation of the expectation is analytically intractable and thus requires numerical approximation. A first approach consists in approximating the expectation by a finite sum over some points sampled from $p(u) \sim \mathcal{N}(0,1)$. It is however more computationally efficient to approximate the integral through the standard Gauss-Hermite quadrature which results in a finite weighted sum and does not require to run a sampling algorithm.

The Gaus-Hermite approximation can similarly be exploited for prediction since the evaluation of the new outputs also require the evaluation of the expectation of a product of probit functions.

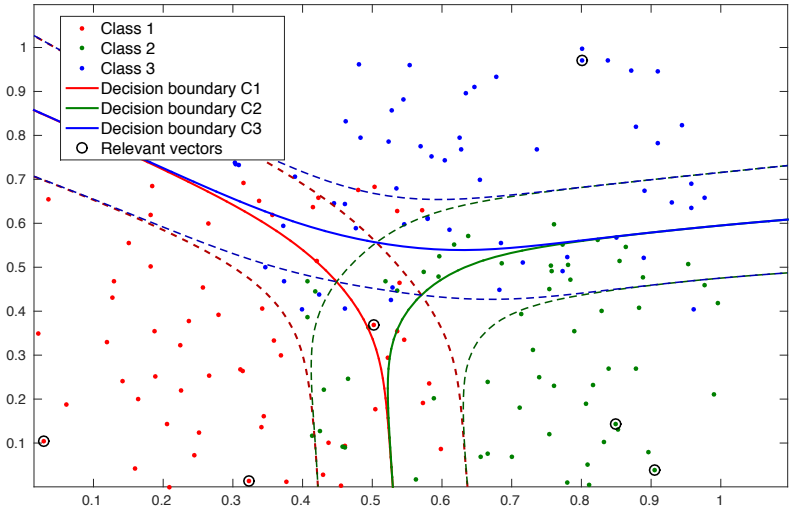


Figure 5.5: Illustration of the mRVM classifier in the 3−class setting. Dashed lines represent the decision boundary *i.e* $p(C_k|x) = 0.5\,^{+}/_{-}\,0.25$ for each class. Note that in this scenario the model selected 6 relevant vector out of 180 samples.

The training procedure is an extension of the usual RVM training method with augmented complexity due to the presence of the new intermediate variable y. Recalling that sparsity arises from the specific type-2 maximum likelihood procedure, this means that we need to infer at first the value of $\mathbf{y}$ before we optimize the intermediate marginal likelihood $p(\mathbf{y}|\mathbf{x}, A)$ where $A \in \mathbb{R}^{N*C}$ denotes the precision matrix of parameters α_{nc}. This naturally leads to the exploitation of the EM algorithm in which the E-step consists in inferring the expected value of y (based on the current posterior over W). In the M-step, we can consequently use the usual RVM optimization method by maximizing $p(\mathbf{y}|\mathbf{x}, \mathbf{A}) = \int(\mathbf{y}|\mathbf{W}, \mathbf{x})p(\mathbf{W}|\mathbf{A})$. Note that this approach allows for the exploitation of the fast training algorithm suggested in (Tipping et al., 2003). An illustration of the mRVM performance over a toy dataset containing 3 classes and a 2−dimensional input is provided in Fig. 5.5.

5.3 RVM based observation selection

We now suggest a new approach for measurement selection that allows for the direct exploitation of standard classification techniques, such as the

mRVM model that is chosen here for its sparsification capabilities. Put in context of the Bayesian framework, the approach we present is approximate as we do not explicitly introduce a hidden variable that models the filters activation weights. Thus, and strictly speaking, this work can be put aside from the rest of the developments suggested in this manuscript, as we purposely ignore the exact Bayesian formalism. As will be shown hereafter, this however allows us to introduce a very simple and efficient method for context-dependent measurement selection that require very few additional developments, in comparison to a basic filter implementation.

Here again, we follow the multiple model approach suggested in chapter 4, and assume that the multiple filters only differ in the subset of measurements they exploit. For now, we also consider that the corresponding noise components are known, and the resulting model can be roughly described by the schematic representation given in Fig. 5.6.

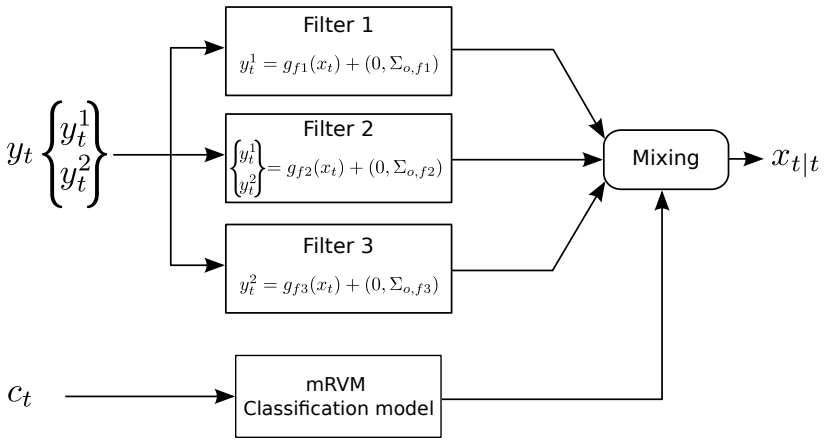


Figure 5.6: Schematic illustration of the mRVM based selection approach for a 2−dimensional observation vector. Recall that c_t denotes the contextual information.

We assume that the observation vector y_t contains N measurements $\{y_t^n\}$. Based on the pre-defined noise free components of the observation model for each of these measurements $\{y_t^n\}$, we can derive the combinatorially exhaustive set of possible observation functions exploiting a different measurement subset. As these functions implicitly denotes the different contexts, we note them f_c where $c \in \{1, ..., C\}$ with C the total amount of distinct functions. We then train a mRVM model such that, at each filter iteration, the model selects the most appropriate subset of measurements $\{y_t^n\}_{n \in c}$ in the complete measurement vector y_t according to the current context, *i.e.* we use the corresponding observation function $f_c(x_t)$:

$$\left[\{y_t^n\}_{n \in [1,...,C]} \subseteq y_t\right] = f_c(x_t)$$

Defining the mRVM training set

The definition of the training data for the mRVM is a crucial step in this approach. The idea is to form an alternative training set based on the collected data $D = \langle \mathbf{x}, \mathbf{y}, \mathbf{c} \rangle$ (respectively the ground truth, the observations exploited for state estimation, and the contextual information) in order to explicitly introduce our requirements in terms of system performance. Thus, each sample $\{x_t, y_t, c_t\}$ in D is associated with an activation vector a_t – *e.g.* $a_t = \{0, 0, 1, 0, ..., 0\}^\intercal$ means that the most appropriate subset at time step t corresponds to the observation function $f_{c=3}$. This yields a new dataset $D' = \langle \mathbf{c}, \mathbf{a} \rangle$ that is used for training a mRVM classifier with C distinct class labels.

At this point, finding the 'most appropriate' subset of measurements requires the evaluation of a score function which relates our expected requirements to the system performance. This score function allows us to chose to train our model generatively or discriminatively. For instance, generative training is achieved by choosing the measurement subset that provides the highest probability $p(\{y_t^n\}_{n\in[1,...,C]}|x_t)$. Note that this does not require to run the recursive equations. Alternatively, discriminative training is achieved by running C filters, each of which exploiting a different function f_c, and then choosing at each time step the filter that provides the most accurate state estimate. Practically, this is done by choosing the filter with the highest probability $\mathcal{N}(x_t|\hat{x}_t^c, \sigma^c)$ where $\hat{x}_t^c$ and σ^c are the output mean and variance of the filter based on the observation function f_c. If only a subset of the components in the state variable x_t is available for training, we can straightforwardly exploit the marginal likelihood (3.8) described in section 3.3.3.

Interestingly, the approach allows any arbitrary score function to be used. In practice, the suggested discriminative approach is equivalent to selecting the filter with the most consistent output. However, if we are mainly interested in reducing the error in the prediction mean, and pay no attention to the accuracy of the output uncertainty, it is possible to exploit the RMS error score (between the filter output and the ground truth). Note that this particular score also allows for the exploitation of a partially observed state variable. Using the notation introduced in section 3.3.3, this means that, at each time step t, we choose the filter providing the lowest error $\parallel h(\hat{x}_t^c) - v_t \parallel$.

In practice, the examination of the filters activation frequencies provided by the classifier on the training set provides useful insights about the relevant measurement combinations. Some filters may appear to be rarely used and therefore can be removed from the initial set before re-training a refined classifier. This process can simplify the classification problem and directly reduce the computational cost of the model at runtime.

Mixing the multiple filters outputs

Until now, we did not give any detail about the method for interfacing the mRVM class label outputs with the multiple filter outputs. In the previous chapter, the mixture of experts framework naturally led to mixing through a weighted sum of each estimate. While this approach is still technically feasible here since the mRVM provides normalized probabilistic outputs (acting as gating weights), it is however inapplicable during training. In the mixture of expert based approach, while running the EM algorithm for training the gating network, it is straightforward to exploit the current model parameters to re-evaluate the gating weights. These weights are used to evaluate the final state belief at each time step, which in turn influences the estimate of the next filter iterations. Consequently, we can train the model by using the exact same inference method that is exploited at runtime.

On the other hand, exploiting a similar approach with the mRVM based approach means that the quality of each filter output will vary at each iteration of the EM algorithm. Consequently the activation vector a_t needs to be re-evaluated, and the new training set D' will vary in time during training. Then there is no convergence guarantee concerning the training algorithm, and generally speaking, the initial assumptions above which the mRVM is built are now invalid. It is clear then that we can not train the model with the mixing method that will be used at runtime. Intuitively, and as described in (Bar-Shalom et al., 2002), an other approximation for mixing multiple state beliefs consists in propagating only the belief with the largest weight and discard the others. In other words, this means that the prior belief used by all the different filters at the next iteration is the one with the highest probability, which in our case corresponds to the filter providing the best consistency score (the local likelihood $\mathcal{N}(x_t|\hat{x}_t^c, \sigma^c)$). This method can thus be used for both the training step, and at runtime.

Note however that, to our knowledge, there is actually no restriction regarding the mixture method we decide to exploit at runtime. In this work however, we find more coherent to train the model based on the inference method that will be used later.

Exploiting non-linear/diverse filters

As for the approach based on the Mixture of Experts framework, we can straightforwardly exploit UKF and EKF filters as they provide Gaussian outputs that allow for the use of recursive equations. In this case, since we do not use GPB mixing, we can also consider the exploitation of particle filters. However, note that we have not analysed the statistical properties of the error introduced by the stochastic approximation in a such a switching model. Thus, and unlike standard particle filters, we have no guarantee that the system will converge to the true state distribution for a large number of

samples.

Since the only constraint about the filters we exploit is to ensure that the state belief they provide satisfy the recursive filtering equations, we can consider to augment the filter bank with a single prediction model. If it seems very unlikely that the best state belief we can obtain is based on the sole dynamic model, we can imagine that some systems may undergo some transient disturbances which for no measurement can temporarily improve the estimation process.

5.4 Experiments

In this section we illustrate the performance of the mRVM based observation selection method on the simulated and real data that was used in chapter 4. We follow the same approach and define the contextual information as a vector containing the raw data provided by the sensors used for state estimation and additional information when available. Recall that, despite its succinctness, it has been shown that this data provides a useful representation of the perception context. The experiments are based on the mRVM code provided by Damoulas[5].

5.4.1 Simulation

As done in the previous chapter, we simulate the data gathered by 3 different sensors providing altitude measurements during a short fly (See Fig.4.7 page 80). The set-up is however different in that we construct 7 observation functions f_c in order to cover the exhaustive measurement combinations: $\{[y^1], [y^2], [y^3], [y^1y^2], [y^1y^3], [y^2y^3], [y^1y^2y^3]\}$. These observation functions are exploited in association with a constant velocity transition model within a simple Kalman filter.

Recall that in this new approach, the final state belief is not evaluated by a weighted sum of the different filters outputs, and is here equal to the estimate provided by the filter with the highest activation probability. Thus, if we want the system to exploit simultaneously multiple sensors, there is no other way than 'forcing' fusion by creating the pre-defined measurement combinations. Note that in the previous chapter, only 3 distinct observation models were built, each of which exploiting only a single sensor, and the simultaneous exploitation of multiple sensors was supported by the gating network. However, due to the peculiar model formulation and the resulting competition between experts, actual sensor fusion was very sporadic in practice.

Based on the 7 original filters, we train a first mRVM model on the 6000 sample dataset corresponding to Fig. 4.7. A direct examination of the filter

[5]Source: http://www.dcs.gla.ac.uk/inference/pMKL/Download.html

activation frequencies on the training set provides useful insight concerning the relevant sensor combinations. In this case the observation functions f_2 and f_5 (exploiting measurement combinations $[y^2]$ and $[y^1 y^3]$) appeared to be exploited on less than 0.1 % of the context samples. The limited exploitation of the sensor 2 is not surprising since it provides the noisiest measurements. Thus, while it permanently provides consistent observations, it is then logical to fuse its measurement with a more accurate sensor in order to reduce the uncertainty. Similarly, it is clear that the combination of the sensor 1 and 3 does not bring many benefits as it would result in a fused information that is inconsistent for both low and high altitudes.

Clearly, the low activation frequency of some observation models does not mean that they play no role in improving the ultimate state belief. For instance, they may play a key role in some very rare occasions that no other observation function f_c can deal with. Thus removing these observation functions from the bank may lead to divergence. In practice, the simplest method for assessing if some observation functions can be removed consists in re-examining the global performance of the refined model. Consequently, we train a new 5-class model and compare its performance with respect to the original model exploiting the 7 measurement combinations. As can be seen in Fig.5.8, the state estimate provided by the refined model on the validation dataset is similar to the original model, whose performance is shown in Fig.5.7. Also, both models provide a similar RMS error (0.035) on the validation dataset.

As can be seen in both figures, the classification approach provides fast switching capabilities, and the transition occurring around 6 meters (when the measurements provided by sensor 1 have to be rejected) is more properly handled than with the mixture based approach (See Fig. 4.8 page 81). Indeed, while mixing outputs through the GPB method provides smooth transitions between filters, it also inherently introduces a short transition 'delay' due to the continuous nature of the gating weights. Note that this phenomenon also depends on the magnitude of the uncertainty in the decision boundaries defined by the gating network, and that we have no control over this parameter.

In Fig.5.9 we show the activation regions of each filter in the 3D space of the context variable defined by the sensor measurements. Note that we must take care to interpret these figures knowing that some parts of this 3-dimensional space do not correspond to any existing conditions (for example the sensor 1 providing a measurement equal to 0 while both other sensors provide measure an altitude of 8 meters). The purpose of this representation is to illustrate the precision of the RVM in the definition of the different activation regions. Moreover, it shows that the new model took advantage of the RVM capability in defining multimodal regions.

For instance, the observation function f_1 (sensor 1 alone) is mostly used in intermediary altitudes (between 1 meter and 5 meters), especially when

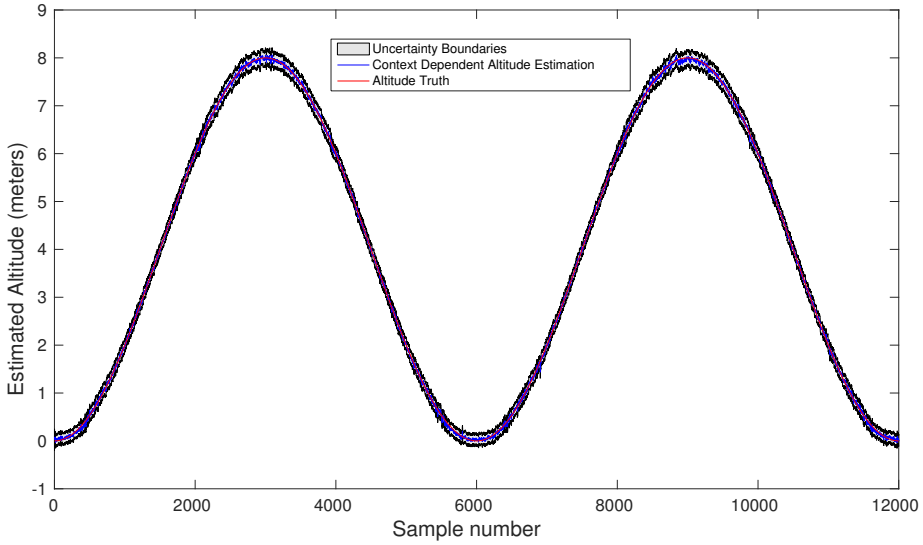


Figure 5.7: Estimated altitude on the validation dataset for the 7-class model.

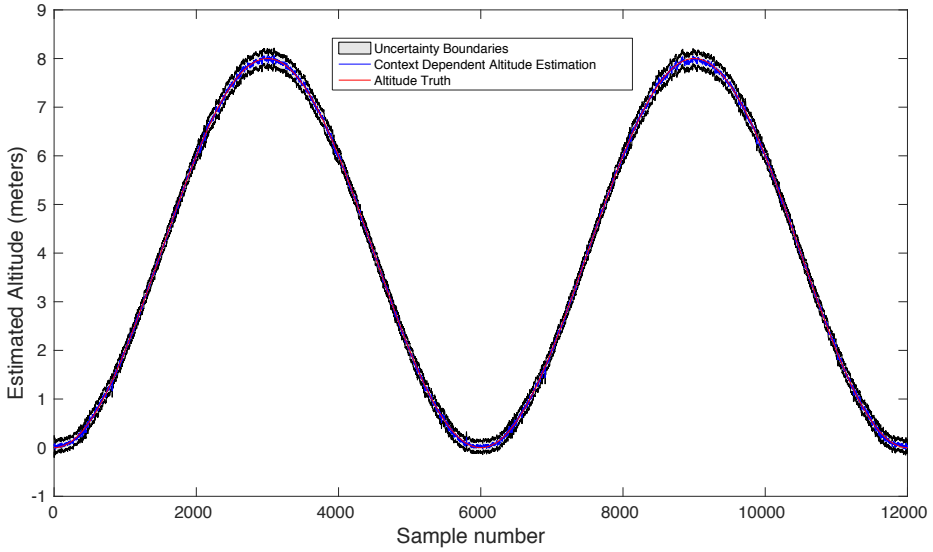


Figure 5.8: Estimated altitude on the validation dataset for the refined model (5 observation functions corresponding to $\{[y^1], [y^3], [y^1y^2], [y^2y^3], [y^1y^2y^3]\}$).

the sensor 3 is still not providing any reliable measure. Note that the same function appears to be activated in some parts of input space that were not seen in the training set. This is the case for $y^1 > 5$ and $y^2 \approx 0$, or for $y^3 = 0$ and $y^2 > 5$. This phenomenon is a side effect of the mRVM approach, which

normalizes the output of each regression model. Thus, while each underlying regression model normally provides a zero centred prediction, the ultimate class membership probabilities are then evaluated based on insignificantly low values, leading to arbitrary decisions.

We find some examples of these arbitrary activation points in all of the 5 distinct activation regions showed in Fig. 5.9. While this makes the interpretation of these figures quite complex, they capture some interesting clues regarding the system behaviour. For instance, we can see in Fig.5.9d that the combination of sensor 2 and 3 is exploited in some different parts of the input space. The first region corresponds to intermediate altitudes, precisely where y^1 is close to zero. This corresponds to the rejection of the outliers provided by the sensor 1. The second region is for high altitudes where both sensor 2 and 3 provide useful information. Similarly, the combination of all sensors is mostly used when all sensors provide a similar measure (center of the cube), and punctually for very low altitudes where all sensors agree on a measure close to zero.

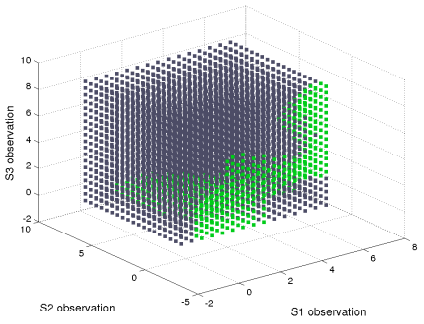
Generally speaking, the classifier recognizes the different contexts which for some sensor measurements could alter the estimation, while combining the maximum of reliable data so as to provide a low estimation uncertainty. In practice, this means that the classifier tries to exploit combined measurements as often as possible. For instance, this leads to a reduced uncertainty in the estimate for altitudes > 6 meters, where the mixture of experts approach was exploiting a single sensor.

Unsurprisingly, training the mRVM is significantly slower in comparison to the mixture of experts approach. Under this setting, the training step takes 15 minutes in average for a dataset containing 5000 points. It however results in a highly sparsified data since only 42 relevant vectors were kept. Consequently, the model is very efficient in making new predictions. In the case of a simple 3-dimensional input space, we observed an average prediction time of 0.002 seconds, based on non-optimized Matlab code.

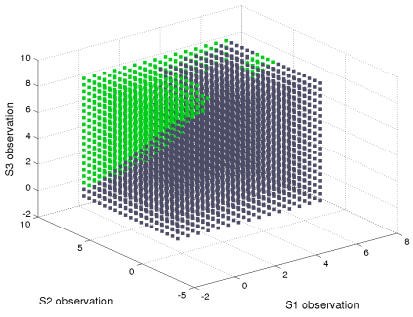
The global performance of the system is also sensitive to the length scale parameter (*cf.* parameter σ in (5.3)) that is here tuned manually. The best value found on this dataset is $\sigma = 0.6$. While other values were usually not decreasing the overall performance of the model, they however frequently led to a slight deterioration of the rejection capability regarding the outliers provided by the sensor 1.

5.4.2 Real data

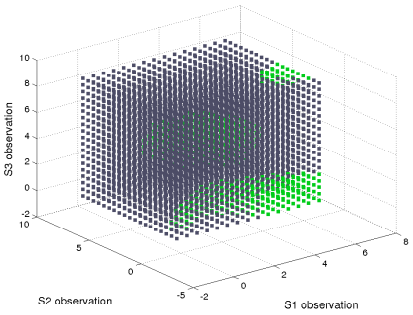
The selection model is now illustrated on real data, using the datasets provided by our paparazzi platform, and used in the previous chapter (See Fig. 4.10 page 83). For comparison purpose, we follow the procedure of section 4.4.2 and use the same training set, before illustrating its performance on the same validation set. The resulting altitude estimate is shown Fig. 5.10.



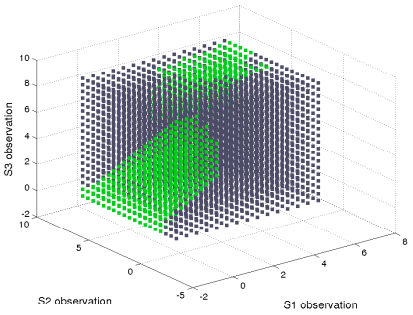
(a) Activation region for $y^1 = f_1(x_t)$.



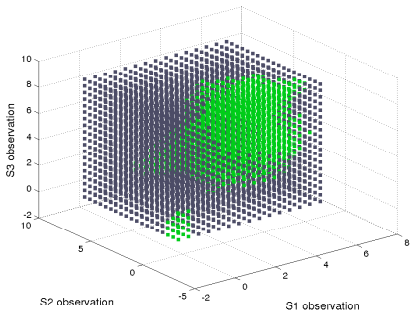
(b) Activation region for $y^3 = f_2(x_t)$.



(c) Activation region for $[y^1 y^2]^\intercal = f_3(x_t)$.



(d) Activation region for $[y^2 y^3]^\intercal = f_4(x_t)$.



(e) Activation region for $[y^1 y^2 y^3]^\intercal = f_5(x_t)$.

Figure 5.9: Activation regions (in green) in the 3-dimensional contextual input space for the 5 filters exploited in the refined model.

As the estimation task is more complex on real data, the resulting mRVM model uses here 126 relevant vectors, but its prediction time remains similar to that of the model we obtained on simulated data (0.002 seconds). As can be seen, most outliers are also avoided here, but we can detect that the system sometimes selected an irrelevant sensor. This is for example the case in the interval between samples 3000 and 3500, where the ultrasonic

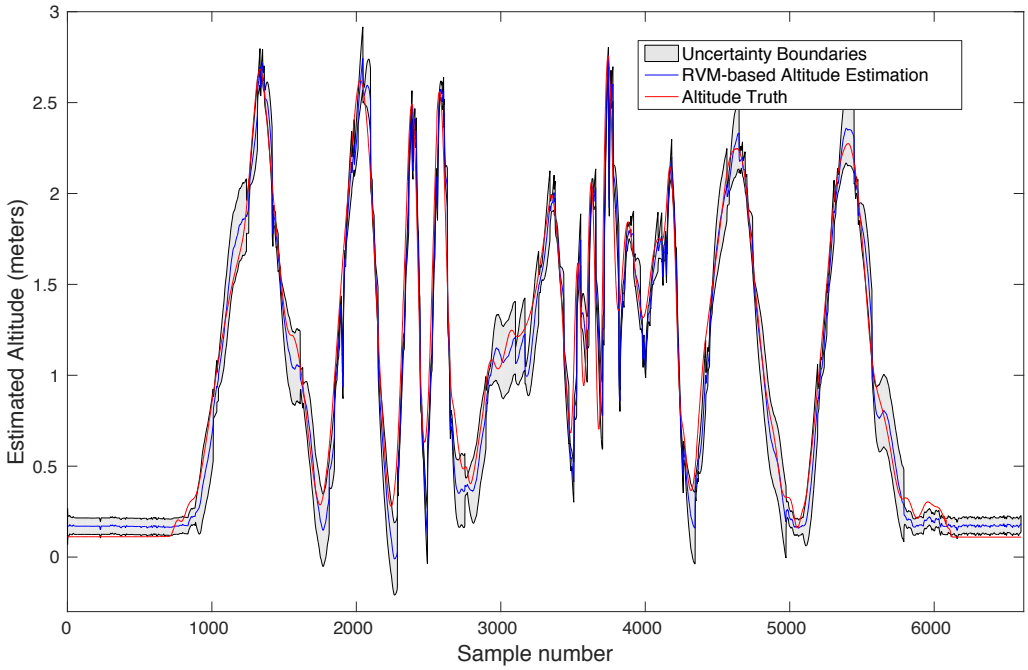


Figure 5.10: Estimated altitude on the validation dataset.

sensor is used, while it presents some typical outliers. From a general point of view, we could not outperform the performance provided by the Mixture of Experts approach in terms of RMS error. Here for example, the mRVM based approach provides an RMS error of 0.152, while we previously obtained a score of 0.135. Note however that this score remains significantly better than for classical rejection.

Clearly, experiments shown that the mRVM model does not generalize as well as the Mixture of Experts. While it may seem surprising given the efficiency of sparse kernel methods, we can identify two main explanations regarding this behavior. First, as said earlier, we did not follow a strict Bayesian approach, and unlike the previous method, we did not introduce a dedicated hidden variable for modeling the activation of each filter. Instead, we directly trained the selection model on an artificial dataset, and the mRVM model subsequently provide predictions that are 'hard' reproductions of what was observed in the training data. Second, the mRVM model provides much more expressiveness than the Gaussian kernels used earlier, and it is well known that the model complexity plays a central role in generalization, as it should match the underlying difficulty of the problem. Consequently, we may overfit the training set as the altitude estimation task does not require to deploy such a complex model. Both these reasons explain why the performance of the mRVM model are slightly lower than

with the Mixture of Experts that combine simplicity and exact Bayesian treatment. However, the approach suggested in this chapter remain more efficient than the classical rejection scheme, and must be tested on more complex scenarios, where the Mixture of Experts based approach might be outperformed. It also provides a simple solution for efficient measurement selection as no specific knowledge of the Bayesian framework is required, and available classification libraries can be straightforwardly used.

5.5 Conclusion and remarks

5.5.1 Summary

In this chapter we presented a generic classification based mechanism for augmenting a Bayes filter with context-dependent observation selection capability. This mechanism intrinsically allows for the straightforward exploitation of any existing classification model. In order to be able to deal with potentially high-dimensional input space and regarding the precision required in defining the decision boundaries, we chose to exploit the non-parametric and sparse Relevance Vector Machine. By combining some specific prior distributions with the type-2 maximum likelihood, this model achieves great sparsity, while automatically adapting the number of relevant vectors to the problem difficulty. We especially believe that, despite its approximate nature regarding the Bayesian framework, the RVM model is a key component for enhancing the standard Bayes filter with online adaptation capabilities.

Here again, we rely on the bank of filters approach where the selection mechanism is ensured by a discriminatively trained model, *i.e* such as providing the best estimate at each time step. The mRVM classification method provides strong sparsification capabilities allowing for fast prediction given new inputs. Also, the mRVM model provides strong genericness regarding the dimensionality and the nature of the input space. Care has however to be taken when the new inputs lie too far from the training set as the class membership and associated uncertainty are then strongly inconsistent.

This approach can be seen as an extension of the model proposed in chapter 4 in that it allows for the definition of arbitrarily complex decision boundaries. Also, the new approach has the advantage of relying on the basic RVM training and inference methods, thus its implementation is simple and allows for the exploitation of available libraries.

5.5.2 Issues

We recall that, as for the mixture based approach, the model proposed in this chapter also assumes that the measurements within the context variable are synchronised. In settings presenting a strong difference between the

contextual measurement frequencies, the selection model may be strongly affected and provide irrelevant outputs.

We noticed that the global system was unsurprisingly sensible to the kernel width parameter. In the current implementation, this parameter is still manually tuned, which represents a non negligible engineering cost when training the model, given the average convergence rate of the EM algorithm (around 15 minutes). Also, the current implementation relies on isotropic Gaussian kernels. While it seems to provide satisfying results in our experiments, some more complex configurations may however require to use distinct width parameters on the different dimensions of the context input space. This would obviously increase the computational cost of training.

One disadvantage of the RVM model is that it usually becomes increasingly certain when extrapolating too far from the training data. While this behavior could still be identified with a basic regression model (since it only requires to detect when the predictive mean goes to zero and the uncertainty converges to a known value) the mRVM output does not present such distinguishable behavior. This is due to the introduction of the multivariate probit regression method which 'hides' this phenomenon, as it provides normalized probabilities through a 1-of-K coding scheme.

5.5.3 Future directions

Enhancing the model expressiveness

During our experimental investigations, we observed that the model generalization capabilities where slightly lower that for the Mixture of Experts approach. As discussed, in the experiments section, this is partly due to the 'hard' assignment of the activation regions we provide in the training set. Also, from a general point of view, adding extra hidden variables in our system representation is known to improve the overall performance as it can capture unknown aspects. Thus, a first solution for improving the model generalization capability would consist in following a proper Bayesian approach, and introduce a gating hidden variable, as done in the previous chapter.

At first glance, it appeared during our experiments that the mRVM model already provides great precision capabilities in defining the activation region boundaries, as we tend to overfit the training data. However, we also discussed the need for defining different kernel width parameters on the different dimensions of the input space. An indirect solution to this problem, and more generally to deal with highly heterogeneous data consists in exploiting the multi-kernel method. This approach basically combines multiple base kernels trough a weighted sum, and has been widely developed within the SVM framework. As shown in (Damoulas and Girolami, 2008),

the mRVM model can straightforwardly be enhanced with multi-kernel capabilities where the Bayesian treatment also leads to sparse solutions in the number of kernels. The resulting model is clearly more complex and comes at an increased computational cost for both training and inference, but proved to achieve state-of-the-art performance on multi-feature and multi-class problems thanks to its genericness.

Learning a single context-dependent observation model

In chapter 3 we discussed the issues arising in methods that generatively learn a regression model for the mapping $x_t \mapsto y_t$. The main issue with this basic formulation is that external phenomenons such as the context are ignored by construction and the resulting model suffers the usual shortcomings when facing real polluted data.

In both mixture based and classification based approaches, we tried to incorporate the state value as an input of the observation selection model, and this always proved to lower the performance of the system. This shows that a state-dependent model trained with the ground truth is more likely to produce erroneous predictions, as errors on the system state are unavoidable at runtime (hence differing from values seen in the training set). Clearly, learning a direct regression model also suffers from this robustness issue.

However, representing the observation distribution through a single model still presents an elegant and efficient solution. In particular, we can replace the bank of models by a single component that would straightforwardly model the mapping $x_t, c_t \mapsto y_t$. Once again, a non-parametric and sparse model such as the RVM would be a perfect candidate for learning this mapping, especially since it provides a Gaussian output that can be directly exploited within a Bayes filter, as suggested in (Ko and Fox, 2009). As discussed, this approach would not be as reliable as the one suggested in this chapter because of the errors produced in the state belief at runtime. A solution for improving robustness regarding this aspect may however consist in assuming that the input data is also corrupted by an unknown noise process. Relaxing the noise free assumption in the input data has been explored within GP regression in (McHutchon and Rasmussen, 2011), and could be adapted to the RVM model in future work.

Note that in our case, discriminative training automatically takes into account the errors made in the state belief at runtime since training involves an iterative re-evaluation of the filter outputs. Thus the model parameters are optimized in accordance with the real noisy system output.

Chapter 6

Context-dependent observation noise adaptation

Contents

Following the modeling paradigm introduced in chapter 3 we now address the context-dependent noise adaptation task (cf. Fig. 3.4 page 56). The extension of the basic state-observation model leads to complex inference for both noise variable and state posterior (*i.e* the update equation). Generative and discriminative approaches are discussed, and we show that the training method influences the complexity of inference and learning in different ways. In both cases, we describe and characterize a set of approximations that allow to exploit the complete system online. Once again, we can interpret these developments as new approaches to adaptive filtering where, unlike the previous selection models, the additional time-heterogeneous variable is now continuous. When the model is trained in a generative manner, we will also see that our approach can be placed in the active research topic that is heteroscedastic regression. This chapter covers in detail the concepts proposed in (Ravet et al., 2014) and (Ravet and Lacroix, 2014) and strongly relies on the theoretical analysis of the RVM regression model provided in the previous chapter.

6.1 Background on time-heterogeneous observation noise models

As discussed in chapter 3, the standard Bayes filters rely on the state-observation model which inherits the assumption of time-homogeneity originally introduced by the Markov chain model. This results in the graphical model depicted in Fig. 6.1 where the static observation noise is represented through the covariance parameter Σ_o (so that $y_t \sim \mathcal{N}(g(x_t), \Sigma_o)$). Most often, the parameter which is considered to be 'optimal' is the one that averagely explains at best a dataset $\big\{\{x_1, y_1\}, \ldots, \{x_n, y_n\}\big\}$. Following the usual modeling strategy, designing the observation model thus consists in finding the best distribution modeling the emission process for the nominal cases, and rejecting all the data that does not fit this distribution (Sivia and Skilling, 2006).

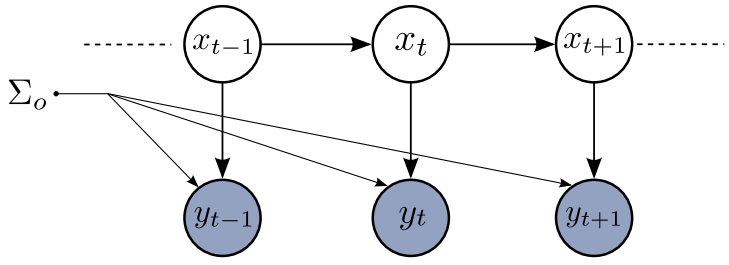


Figure 6.1: Graphical model of a state-observation model with homogeneous observation distributions. Note that the stationary noise covariance is not represented by a random variable.

For many applications however, and in robotics especially, there is no stationary observation noise value that would yield an optimal output for a large variety of operating conditions (or environments). This problem motivated the development of adaptive filtering techniques (Mehra, 1972) where the noise parameters are continuously estimated with the system state. Some original approaches for adaptive filtering rely on an online analysis of the model statistics. For instance the covariance matching method adapts the noise parameters so that the actual (measured) covariance of the filter innovation is consistent with its theoretical value. More general is the Bayesian approach that yields the multiple model method as well as the state augmentation methods in which the model parameters are integrated as part of the system state.

The multiple model approach can obviously give a first answer for modeling systems with time-varying observation noise (Bar-Shalom et al., 2002; Ghahramani and Hinton, 1998). In that case we could straightforwardly exploit the techniques presented before in the context of measurement selection. However, unlike measurement selection where the model intrinsically switches between a finite number of observation functions, assuming that the noise behavior can be segmented into multiple regimes is irrelevant in a real scenario. Recall that our main objective in this work is to model the context influence over the observation model without introducing any expertise. Furthermore, for most practical applications, the physical phenomenons involved in the context influence over the measurement process remains unexplained.

Examples of state-observation models in which Bayesian inference is recursively done for both the system state and the observation noise are suggested in (Sarkka and Nummenmaa, 2009; Agamennoni et al., 2011). These articles provide a sound analysis based on the variational approximation for evaluating the posterior distribution $p(x_t, \Sigma_{o_t}|y_{1:t})$ through a decoupled product of densities for x_t and Σ_{o_t}. The corresponding graphical model is depicted in Fig. 6.2. The estimation of the noise characteristics is then done in an unsupervised manner, and the method potentially suffers the shortcomings discussed in chapter 3. Furthermore, these models do not allow for the introduction of an external input that would govern the noise parameter. In practice, their improvement regarding the standard LDS performance is mainly due to the choice of an adequate prior distribution over Σ_o which results in an outlier robust observation distribution. One interesting remark concerning the model depicted in Fig. 6.2 is that the noise parameter Σ_{o_t} and the state x_t are conditionally dependent through the active trail $\Sigma_{o_t} \rightarrow y_t \leftarrow x_t$, hence yielding more complex inference.

At the same time, an alternative approach has been suggested in (Ko and Fox, 2009) along with the introduction of non-parametric regression methods

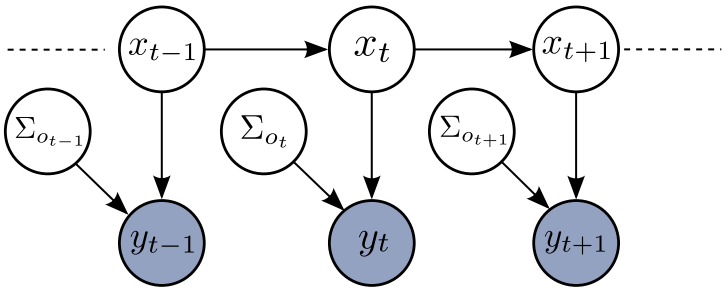


Figure 6.2: Graphical model of a state-observation model with heterogeneous observation distributions.

(Deisenroth et al., 2009; 2012) for learning the prediction and observation distribution of a state-observation model. This led to the previously discussed heteroscedastic observation model of the form $y_t = g(x_t) + \epsilon(x_t)$ where both g and ϵ are given by non-parametric models such as GPs. This augmented form of regression is illustrated in Fig. 6.3 where a basic GP is compared to the heteroscedastic method described in (Kersting et al., 2007).

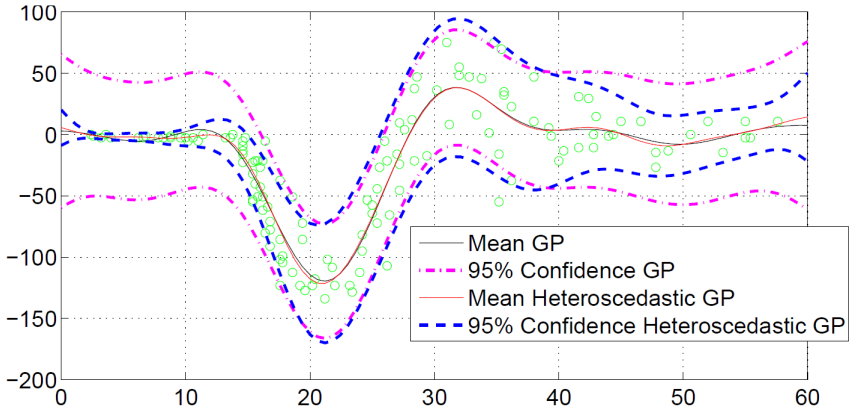


Figure 6.3: Comparison of standard GP regression model with heteroscedastic GP regression. The standard GP model provides accurate prediction mean but the output variance is sometimes inconsistent with the noise in the data, due to the assumption of constant noise. On the other hand, heteroscedastic regression takes into account the noise observed in the data and provides a consistent output uncertainty.(Image excerpt from (Kersting et al., 2007))

As shown in Fig. 6.3, heteroscedastic regression naturally provides input dependent noise adaptation capabilities, as the their prediction variance stays consistent with the noise observed in the training data. When used within a Bayes filter, this method can greatly improve the consistency of

the state estimate. However, two important aspects are not discussed in the work of (Ko and Fox, 2009). First, since the observation distribution is fully state-dependent, the resulting model is particularly sensitive to the presence of outliers, which may initiate a cycle of error propagation. This is because outliers are usually not modeled as such by the observation distribution. Subsequently, unless similar outliers occur for the exact same state value in the training set and at runtime [1], strong errors in the state estimate now results from two major issues. At first, the filter is not able to process 'new' outliers, as any standard Bayes filter. Second, the mapping $x_t \mapsto y_t$ may encode the presence of outliers seen in the training set in place of the theoretical correct observation. Subsequently, errors in the state estimate lead to poor performance of the regression model which, we shall recall, is not robust to noisy inputs, and filter divergence is more likely to happen. To illustrate this problem, we trained two GP observation models on the real dataset exploited in the previous sections (one for each sensor), and run a GP-BayesFilter on a validation set following (Ko and Fox, 2009) using the GP library from Rasmussen and Williams[2]. The resulting output is shown in Fig. 6.4 where we observe multiple cases of divergence. Note that divergence always occur over a limited amount of time as the altitude profile allows the filter to 'catch up' with the correct estimate.

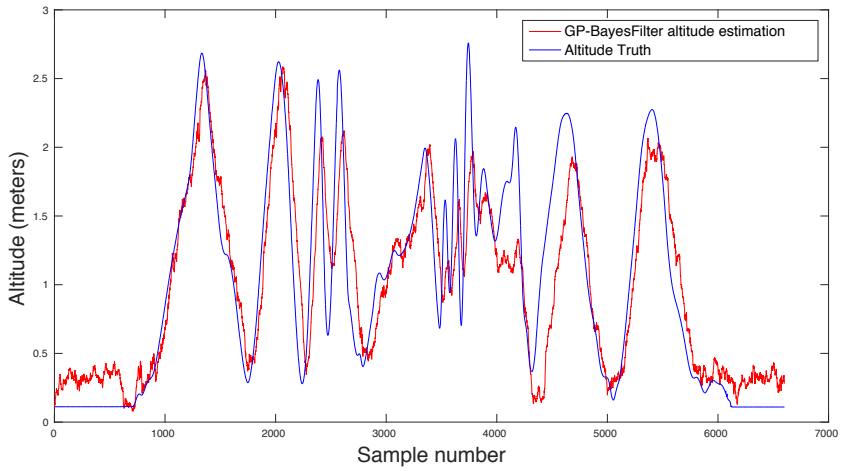


Figure 6.4: Altitude estimation provided by a GP-BayesFilter on a real validation set. Multiple cases of divergence can be observed, where the state belief evolution is mainly governed by the dynamic model $p(x_t|x_{t-1})$.

The second problem that is not discussed in (Ko and Fox, 2009) concerns inference in the augmented state-observation model. In this approach,

[1] In which case we can not consider these measurements as outliers
[2] Source: http://www.GaussianProcess.org/gpml/code

learning a mapping $x_t \mapsto y_t$ is treated as a standard heteroscedastic regression problem. This purely generative approach is interpreted as the standard construction step in which we define the observation model. Then, this distribution is straightforwardly exploited through the usual recursive filtering equations. This is ignoring the structure of the new underlying Bayesian network, which for the standard prediction method for evaluating the distribution $p(y_t|x_t, \Sigma_{o_t})$ is now an approximation. More precisely, the interdependency between the observation noise and the current state depicted in Fig. 6.2 is ignored. Intuitively, this means that we run inference on two separate models, without taking into account the self-consistency of the noise variable with the state-observation model. Thus, exploiting the standard recursive equations is an inexact form of inference in this case. Note however, that avoiding to rely on self-consistency was one of our core motivation for the development of new approaches. Furthermore, while it does not respect the exact Bayesian treatment, this approach proved to perform very well.

In the next section, we derive a new heteroscedastic, context-dependent observation model that aims at dealing more properly with the issues emerging from the regression approach as used in (Ko and Fox, 2009). This is done through the introduction of contextual information, but also with the help of a new modeling paradigm. Computational efficiency is here again ensured by the exploitation of the RVM model that we introduced in the previous chapter. In this discussion, we will also try to keep a clear understanding of the approximations done when running inference in the model. Interestingly, we will see that discriminative training can help in improving this specific aspect.

6.2 Sparse Bayesian models for context-dependent observation noise

6.2.1 Building a new observation model

As done in the previous sections, we overcome the standard Bayes filter unability to deal with context influence by augmenting the state-observation model with a new component. Here, the role of this component is to explicit the context influence over the observation noise. For this purpose, we still rely on an additional observation variable c_t relating to the perception context. To further avoid ambiguities in the contribution of the two distinct observation components, *i.e* the deterministic noise-free observation function g and the noise model ϵ, we assume that g is known and time-homogeneous. Note that the observation function can generally be obtained directly through physical considerations about the nominal measurement

generation process. This results in an observation model of the form:

$$y_t = g(x_t) + \epsilon(c_t)$$

Reminding that we consider here systems which for the assumption of Gaussian noise is valid, $\epsilon(c_t)$ is then represented by a zero mean Gaussian noise distribution with context-dependent variance:

$$\epsilon(c_t) \sim \mathcal{N}\big(0,\, r(c_t)\big)$$

To avoid making any assumption over the functional form for the variance model, and keep computational efficiency, $r(c_t)$ is modeled by a RVM regression model, naturally providing sparsity thanks to the ARD mechanism. For a given training set of T observations $\{c_t\}_{t=1}^{T}$, we define $r(c_t) = exp(z_t)$ to ensure variance positivity where z_t is given by

$$z_t = \sum_{i=1}^{T} w_{i+1} K(c_t, c_i) + w_1 + \nu \tag{6.1}$$

with w_1 the bias parameter, K the chosen kernel function, and $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ the component representing the noise in the data. Following the usual RVM formulation, an independent zero-mean Gaussian prior is placed over each component of the weight vector $\mathbf{w} = (w_1, .., w_{T+1})^\intercal$ so that

$$p(w_i|\alpha_i) = \mathcal{N}(w_i \,|\, 0, \alpha_i^{-1})$$

where we define uniform distributions over each hyperparameter α_i (Recall that this specific prior fosters sparsity when associated with type-2 maximum likelihood).

Exploiting multi-dimensional observations

So far, the model has been depicted for a 1-dimensional observation space. Real applications however require to consider the multi-dimensional case. In these first developments, we assume that for an observation variable $y_t \in \mathbb{R}^D$, the associated noise covariance denoted $\mathcal{E}(c_t)$ is diagonal. This assumption is principally made as it reduces the amount of additional variables in the model. Furthermore, as a consequence, it implicitly forces the selection component described in the previous chapters to be responsible for explaining the correlation between the measurements. The original RVM regression model (Tipping, 2001) uniquely allowing for mappings from a multivariate input to a univariate output, we need to rework the model for the multi-dimensional case. One simple solution is to exploit D independent models for each component of the noise covariance. This however means that we have to exploit distinct sets of relevance vectors for prediction, yielding additional computations. Alternatively, the RVM regression

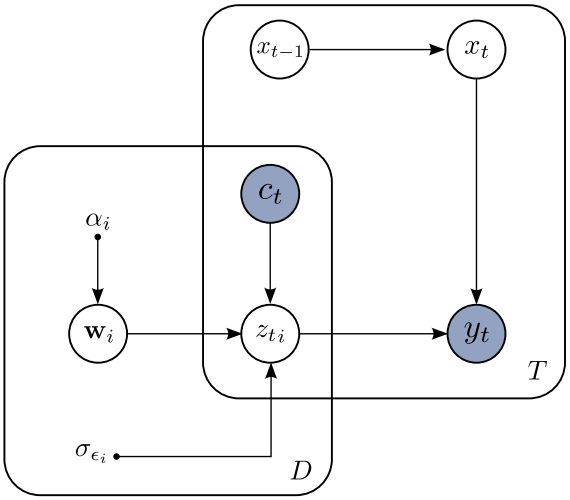


Figure 6.5: State-observation model augmented with a RVM based context-dependent noise component. Note that we use a template-based representation where T denotes the number of samples in the training set, and D denotes the dimensionality of the observation vector y_t.

model can be extended to treat multivariate outputs while exploiting a common set of relevance vectors. A detailed discussion of this approach can be found in (Thayananthan et al., 2006; Thayananthan, 2005). For clarity, the next discussion only considers the one-dimensional case, the extension to higher dimensions being straightforward. The graphical representation of the augmented state-observation model is depicted in Fig. 6.5.

6.2.2 Generative training

As assumed so far in our discussion, the following approach requires that we are given a training set $\mathcal{D}$ containing the ground truth and their associated observation, so that $\mathcal{D} = \{x_t, y_t, c_t\}_{t=1}^{T}$. Recall that in the context of generative training, only the subset of variables in x_t that are actually exploited by the observation model are required. Using d-separation on the graph depicted in Fig. 6.5, we see that the joint distribution over the system variables factorizes as follow:

$$p(\mathbf{x}, \mathbf{y}, \mathbf{c}, \mathbf{z}, \mathbf{w}) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1}) \prod_{t=1}^{T} p(y_t|x_t, z_t) p(z_t|c_t, \mathbf{w}) p(\mathbf{w}|\alpha)$$

where $p(x_1)$ denotes the initial state belief, defined here by a Gaussian distribution of known mean and covariance. We can notice the similarity between this decomposition and the one exploited in the standard generative training of the state-observation model (Eq. (3.5)). The similarity resides in the

separation of the training task for the prediction model and for the observation model. Practically, the issue of training the new observation model turns to be analogous to the heteroscedastic regression task, which has been discussed in the context of nonparametric models in Goldberg et al. (1998); Kersting et al. (2007); Lázaro-gredilla and Titsias (2011); Khashabi et al. (2013); Ko and Fox (2009); Le and Smola (2005). Note however that the training task is here a bit simpler since the observation function g is fixed and we only focus on learning a model for the noise magnitude $N(c_t)$. As such, the approach we describe in this chapter can be seen as a general purpose regression method that is especially designed to deal with scenarios where the usual heteroscedastic models are mislead by the presence of erroneous observations in the training set. As seen earlier, the later approach is unable to distinguish which part of the observations should be assigned to the observation function g, and which part should be preferably encompassed within the noise component (cf. Fig. 3.3 page 54).

Hybrid EM/type-2 maximum likelihood training

The heteroscedastic regression problem substantially increases the complexity of inference and learning (by comparison to the homoscedastic regression task) due to the introduction of a new hidden variable, *i.e* the varying noise. Many approaches rely on sampling or variational EM algorithm for approximating the posterior distribution of the noise. In our case, the specific RVM model, and more precisely the sparsity arising from the type-2 maximum likelihood, motivates the exploitation of a hybrid method for dealing with the two unobserved variables z_t and $\mathbf{w}$.

Since the standard RVM model provides sparsity when the weight variable $\mathbf{w}$ is marginalized out through the type-2 maximum likelihood[3] method, we shall keep a similar approach. The reminding hidden variable z_t can then be treated through the more conventional EM algorithm. Precisely, we chose to exploit a hard-assignment EM algorithm as suggested in Kersting et al. (2007). Practically, this means that the noise variable z_t is approximated by a single value corresponding to the *most-likely* noise value. As described hereafter, this approximation allows us to make direct use of the standard RVM optimization and prediction equations, and provides in this context the fastest solution for a real time application.

Following the approach suggested in (Kersting et al., 2007), the hard E-step consists in empirical estimation of the noise variance. To empirically evaluate the noise level for each input pair $\{x_t, c_t\}$, we start by sampling K observation samples y_t^k from the current observation model $g(x_t) + \epsilon(c_t)$. Then we consider each pair of values y_t, y_t^k as independent noisy observations of the underlying noise-free observation model $g(x_t)$ so that we can evaluate

[3]Recall that type-2 maximum likelihood is a semi-Bayesian method

the local noise variance through the term $(0.5\,(y_t - y_t^k)^2)$. Note that this term corresponds to the unbiased empirical variance estimate (Bishop, 2006, p.27) and aims at correcting the error introduced by the small amount of data points (overfitting). Taking the expectation of the noise value respectively to the RVM parameters (found after the last EM iteration), we then evaluate the empirical variance at the input $\{x_t, c_t\}$ through the term

$$var_t = \frac{1}{2.K} \sum_{k=1}^{K} (y_t - y_t^k)^2$$

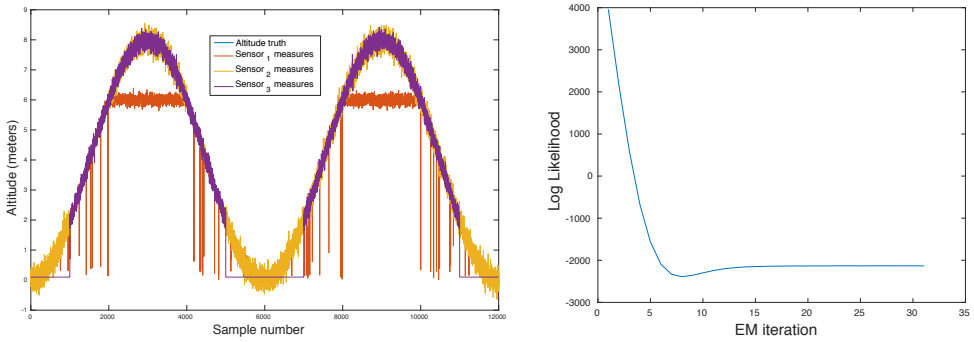
In the subsequent M-step the RVM model is optimized with the new training set $D = \{c_t, log(var_t)\}_{t=1}^{T}$ using the classical optimization procedure (Tipping, 2001). This provides new parameter estimates α^{new} and σ_ν^{new} which are subsequently exploited for sampling in the next E-step. The whole procedure is stopped under the condition that the relevance vectors found after RVM optimization are similar to the one returned at the previous step, and that the change in the parameter α^{new} and σ_ν^{new} is below a fixed convergence threshold. To summarize, the optimization process considers the noise variance as the hidden variable of the model, and iteratively optimizes the parameters of the RVM model following the type-2 maximum likelihood, based on a hard assignment of the estimated noise. The intuition behind this optimization scheme is that each iteration of the EM algorithm decreases the empirical distance between the observations predicted by the model and the 'real' observations y_t provided in the training set.

This method requires to use a substantial number of samples to empirically estimate the noise variance and, as any hard-assignment EM, is prone to oscillating (Kersting et al., 2007), requiring to monitor the likelihood of the model over the training set after each algorithm iteration. It however brings an important advantage, since the optimized model, in association with the last noise variance estimation, can be readily used for prediction using classical RVM equations. Recall that the standard RVM prediction equation (5.10) for regression indeed exploits the output values seen in the training set (In the Bayesian treatment of regression, the output sequence seen in the training set is involved in the evaluation of the posterior distribution over the weight variables $\mathbf{w}$).

Finally, we shall point out that this specific hybrid training method is an unusual and *ad hoc* approximation in the context of the Bayesian framework. Precisely, the EM procedure requires to evaluate the expectation of any hidden variable in the model, while we marginalize out the weight variable $\mathbf{w}$ of the system. This procedure is chosen on purpose as it provides us with the sparsification guarantees of the ARD technique applied to RVM.

Illustration

We illustrate the new observation model behavior on the simulated data exploited earlier in the context of altitude estimation. In this first experiment, we learn the noise prediction model of a sensor providing altitude measurement with time-varying noise (corresponding to the sensor 2 in Fig. 6.6a). The context information is still defined by the 3-dimensional vector containing the raw sensor measurements, and shown in Fig. 6.6a. The log-likelihood progression during the EM training procedure is also illustrated in Fig. 6.6b.



(a) Sensor measurements providing the context information.

(b) Log-likelihood progression during training.

Figure 6.6: Simulated training set and convergence progression of the hybrid training algorithm.

As we can see, the hard-EM procedure rapidly converges to a local minimum, as usually observed with the hard-assignment approximation (Koller and Friedman, 2009, p.885). This behavior contrasts with the exact EM procedure and, loosely speaking, can be justified by the fact that the optimization makes discrete steps in the parameter space. On the other hand, the exact EM procedure follows a continuous path where each iteration changes the parameter value by a small increment. Note that the hard-EM tendency to oscillate after convergence is here worsened by the concurrent type-2 optimization. Indeed, as some relevance vectors are removed along training iterations, the model ability to explain the observations contained in the training set vary, as the generated samples y_t^k directly depend on the current set of relevant points. Subsequently, the hard assignment made over the noise distribution may lower the model likelihood. This phenomenon can be observed in Fig. 6.6b after the local minimum was reached at iteration 8. This convergence behavior was frequently observed in our experiments, where the decrease in the likelihood was usually caused by underestimation of the noise magnitude. In other words, the predictions done by the model

became too close to the observations in the training over the multiple EM iterations.

The resulting noise model, in association to the noise free observation component, is then used to make prediction on the training set. The predicted observation mean and variance are shown Fig. 6.7.

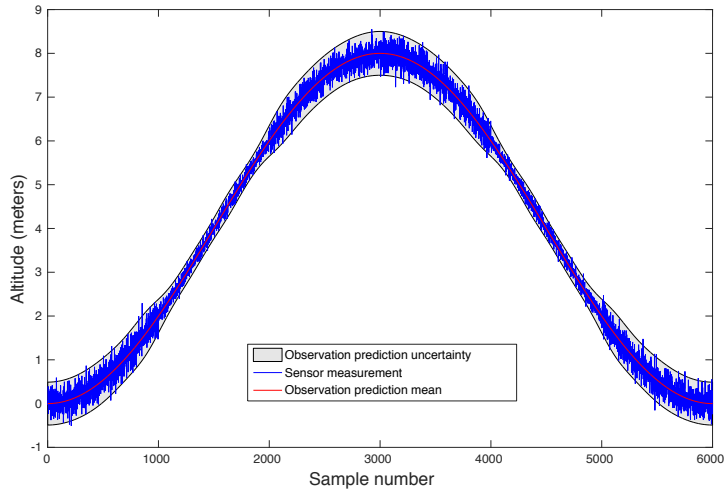


Figure 6.7: Observation predictions $p(y_t|x_t, c_t)$ obtained after 30 EM iterations.

As can be seen, the resulting model shows accurate noise prediction capability, with a slight tendency to underestimation. Note that the training was stopped after 30 iterations, thus the model does not exploit the parameters corresponding to the local minimum shown in Fig. 6.6b. In Fig. 6.8, we show the observation prediction obtained after stopping the training procedure at the lowest log-likelihood value. As can be seen, the resulting observation uncertainty is more consistent with the actual noise in the measurement. It is thus really important to track the progression of the model likelihood during training, since the EM convergence properties are not guaranteed in this approach.

However, and as explained before, one of the main benefits of this hybrid approach resides in the direct exploitation of the ARD mechanism. In this experiment, the type-2 maximum likelihood provided great sparsity since only 11 relevant vectors were kept among the 6000 samples of the training set. Clearly, the resulting computational cost is compliant with online exploitation of a Bayes filter.

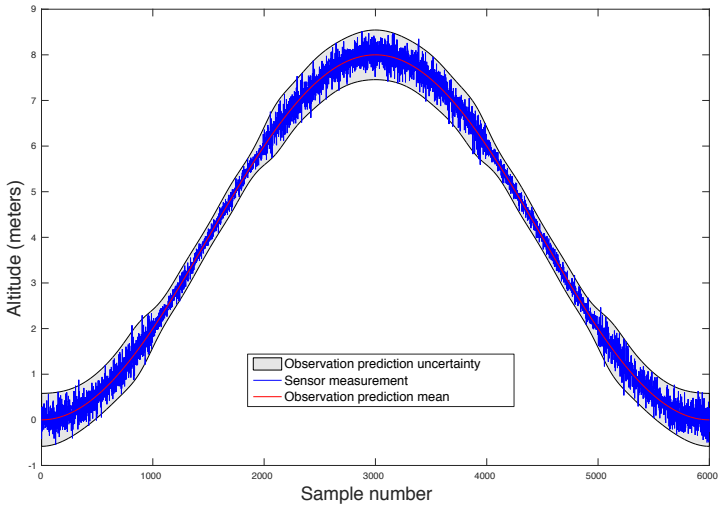


Figure 6.8: Observation predictions $p(y_t|x_t, c_t)$ obtained after 8 EM iterations (log-likelihood local minimum). The observation uncertainty is slightly increased in comparison to Fig. 6.7. This is particularly noticeable in the interval between samples 1000 and 2000, and between samples 4000 and 5000.

6.2.3 Discriminative training

The previous approach aims at minimizing a loss function corresponding to the observation likelihood. In other words the optimization step finds model parameters explaining at best the measurement generation process. As suggested in chapter 3, optimizing the parameters with respect to the ultimate system performance, *i.e* the accuracy of the state estimates, proved to lead to better consistency in the estimation. Training the model then consists in finding the parameters α_{max} and $\sigma_{\nu max}$ such that:

$$\langle \alpha_{max}, \sigma_{\nu max} \rangle = \underset{\alpha,\sigma_\nu}{\operatorname{argmax}} \sum_{t=1}^{T} log\big(p(x_t|y_{1:t}) \big)$$

where $p(x_t|y_{1:t})$ is provided by the recursive equations (3.1).

Since this distribution requires the evaluation of two latent variables, *i.e* the log noise level z_t and the RVM weight vector $\mathbf{w}$, a procedure similar to type-2 maximum likelihood is employed. Note that the EM procedure used earlier is here irrelevant since we automatically remodel the graph structure into its equivalent discriminative representation. Sampling from the generative observation model is consequently unsuitable since the purpose of the z_t variable is not to explain at best the observations y_t based on the noise-free component $g(x_t)$. Following the type-2 approximation, the variables z_t

and $\mathbf{w}$ are then marginalized out and we now seek for parameters α_{max} and $\sigma_{\epsilon max}$ maximizing:

$$\mathcal{L}_{discr} = \sum_{t=1}^{T} log\Big(\int \int p(x_t|y_{1:t}, z_t) p(z_t|c_t, \mathbf{w}, \sigma_\nu) p(\mathbf{w}|\alpha) \, \mathrm{d}\mathbf{w} \, \mathrm{d}z_t \Big)$$

Since $p(z_t|c_t, \mathbf{w}, \sigma_\nu)$ and $p(\mathbf{w}|\alpha)$ define a linear Gaussian model, the integral with respect to $\mathbf{w}$ is readily evaluated to give (*cf.* Eq. (5.7)):

$$\mathcal{L}_{discr} = \sum_{t=1}^{T} log\Big(\int p(x_t|y_{1:t}, z_t) \, \mathcal{N}(z_t|0, D_t) \, \mathrm{d}z_t \Big)$$

where $D_t = \sigma_\nu + K^\intercal A^{-1} K$, with K the vector of kernel functions such that $K_i = K(c_t, c_i)$ as defined in (6.1), and $A = \mathrm{diag}(\alpha)$.

The remaining integral over z_t is analytically intractable and requires approximation. Furthermore, the most likely point approximation exploited in the generative case can not be exploited here as the resulting distribution over z_t is a zero-centred Gaussian. Consequently, any variation in the parameters α and σ_ν would not affect the value of the approximate integral. Alternatively, the integration rule (B.1) exploited in the unscented transform provides an accurate and computationally efficient solution. Note that in the one-dimensional case, this corresponds to the Gaussian quadrature method.

The parameters α_{max} and $\sigma_{\nu max}$ are then determined through conjugate gradient ascent over $\mathcal{L}_{discr}$. Note that classical optimization of the RVM model requires the computation of a *design matrix* containing all kernel elements evaluated at all original locations $\{c_i\}_{i=1}^{T}$. Here, the optimization is done separately for each kernel vector evaluated at c_i, through their influence over ultimate filter performance. The resulting optimization scheme then differs from the standard RVM training. Moreover, the analysis of sparsity provided in appendix C does not hold, and the associated fast training algorithm (Tipping et al., 2003) is not exploitable. Thus the discriminative training approach requires to foster sparsity by thresholding of the α_i values during the optimization process. This approach was also suggested in (Tipping, 2001) in the case of general Gamma priors over the hyperparameters α. We can however decrease the resulting training time by initializing the gradient ascent procedure with the parameters found after a previous generative training step.

6.2.4 Inference in the generative model

In this subsection we start with a brief description of a fast and approximate inference method in the augmented state-observation model depicted in Fig.

6.5. The approximation is due to the separation of inference in two distinct steps, that may appear intuitive, but however inexact given the dependencies encoded in the graph. In the first step, we predict the observation noise using the RVM model, and the context information input c_t. Then, the recursive filtering equations are used, based on the resulting observation model. This inference method is usually performed (Ko and Fox, 2009; Deisenroth et al., 2009; 2012) ignoring that the observation noise variable z_t and the state are coupled through the active trail $x_t \rightarrow y_t \leftarrow z_t$. In practice, exact inference over the model would require to evaluate the posterior distribution $p(x_t, z_t|y_{1:t}, c_t)$, then making the recursive filtering equations inapplicable. Following the fast approximation method, a proper Bayesian analysis of inference is thus given, aiming for a better consistency with the actual dependencies in the model. This analysis yields new recursive equations for estimating both the state and noise variables.

Fast approximate inference

Recall that standard filtering over the state-observation model consists in evaluating the normalized marginal distributions $\hat{\alpha}(x_t) = p(x_t|y_1, .., y_t)$ with the recursion equation

$$\eta_t \hat{\alpha}(x_t) = p(y_t|x_t) \int \hat{\alpha}(x_{t-1}) p(x_t|x_{t-1}) \mathrm{d}x_{t-1} \tag{6.2}$$

In our new model, the noise level is represented by an additional latent variable z_t, and the observation distribution required for the evaluation of (6.2) is now given by:

$$p(y_t|x_t, c_t) = \int \mathcal{N}\big(y_t|g(x_t), exp(z_t)\big) p(z_t|c_t, C, \mathbf{z}, \alpha, \sigma_\nu) \mathrm{d}z_t \tag{6.3}$$

where $\mathbf{z}$ is the predicted log noise level at original inputs $\{c_i\}_{i=1}^T$ (training set). As done for the standard RVM regression model, the predictive distribution $p(z_t|c_t, \mathbf{z}, \alpha, \sigma_\epsilon)$ is evaluated through marginalization of the posterior distribution of the weight variable $\mathbf{w}$:

$$p(z_t|c_t, \mathbf{z}, \alpha, \sigma_\nu) = \int p(z_t|c_t, \mathbf{w}, \sigma_\nu) p(\mathbf{w}|\mathbf{z}, \alpha, \sigma_\nu) \mathrm{d}\mathbf{w} \tag{6.4}$$

This familiar predictive distribution is also Gaussian, as shown by the result 5.10. The evaluation of the integral (6.3) is hence analytically intractable, and requires approximation. The fastest approach is the most likely approximation, where we replace the integral by $\mathcal{N}\big(y_t|g(x_t), exp(z_t^*)\big)$ with $z_t^* = \mathrm{argmax}_{z_t}\, p(z_t|c_t, \mathbf{z}, \alpha, \sigma_\nu)$. Alternatively, the integral can be approximated through the unscented transform, resulting in a set of sigma points that can be used to evaluate the mean and variance of the observation distribution. Note that both approximation methods result in a Gaussian distribution, thus allowing for the exploitation of the standard filtering methods in a state-observation model with assumed Gaussian noise.

Proper Bayesian analysis

Formally, the recursive equations used in the previous approach are not valid within the graph depicted in Fig. 6.5. By ignoring the coupling between the noise variable z_t and the state x_t induced through the distribution $p(y_t|x_t, z_t)$, we leave the noise magnitude to be uniquely governed by the RVM model. This is in accordance with one of our initial motivations, *i.e* avoiding to use model self-consistency as the only score for assessing the measurements reliability. However, and generally speaking, in any state-observation model whose prediction and/or observation distributions are evolving in time, inference from an exact Bayesian perspective requires the joint evaluation of the state and the parameter variables through the same procedure.

In our model, the interdependency between x_t and z_t is induced by the generative representation of the augmented state-observation model which tries to explain at best the observation y_t produced in a given state x_t. Thus, inference over the hidden variables consists in finding the joint set of state and model parameters that explain the observation y_t, in accordance with the the generative representation. Note that in the standard state-observation model, inference consists only in finding the state variable that explains at best the generation of the observation y_t.

In the following discussion, we consider that the posterior distribution over the weight parameter $\mathbf{w}$ is known and given by the standard equation (5.6). Note that marginalizing out this variable in the model is coherent with the type-2 maximum likelihood procedure used in the training step. Consequently, by analogy with the standard state-observation model, filtering now consists in evaluating the joint posterior distribution $p(x_t, z_t|y_{1:t}, c_{1:t})$. However, recall that a pure Bayesian analysis would require to evaluate $p(x_t, z_t, \mathbf{w}|y_{1:t}, c_{1:t})$.

Departing from the previous analysis provided in chapter 3, we now denote the distribution of interest $\alpha(x_t, z_t) = p(x_t, z_t, y_{1:t}, c_{1:t})$ and derive a new recursive equation starting with the following decomposition:

$$\begin{aligned}
\alpha(x_t, z_t) &= p(x_t, z_t, y_{1:t}, c_{1:t}) \\
&= p(y_{1:t}|x_t, z_t, c_{1:t})p(x_t, z_t, c_{1:t}) \\
&= p(y_{1:t-1}|y_t, x_t, z_t, c_{1:t})p(y_t|x_t, z_t, c_{1:t})p(x_t, z_t, c_{1:t}) \\
&= p(y_{1:t-1}|x_t, c_{1:t-1})p(y_t|x_t, z_t)p(x_t, z_t, c_{1:t})
\end{aligned}$$

where the last expression is obtained by exploitation of d-separation properties in the graph. As for the standard state-observation model, the past observations up to time $t-1$ are indeed independent of y_t, z_t and c_t given x_t. Similarly, there is no active trail between y_t and the context observations $c_{1:t}$ given x_t and z_t. The posterior over x_t and z_t can then be computed as

follows:

$$\begin{aligned}\alpha(x_t, z_t) &= p(y_{1:t-1}|x_t, c_{1:t-1})p(y_t|x_t, z_t)p(x_t|z_t, c_{1:t})p(z_t, c_{1:t}) \\ &= p(y_{1:t-1}, x_t|z_t, c_{1:t})p(y_t|x_t, z_t)p(z_t|c_{1:t})p(c_{1:t}) \\ &= p(y_{1:t-1}, x_t|c_{1:t-1})p(y_t|x_t, z_t)p(z_t|c_t)p(c_{1:t})\end{aligned}$$

where once again, we exploit d-separation to obtain the last equality. We can now identify some equivalent prediction and update steps by extracting the prediction distribution $p(x_t|x_{t-1})$:

$$\begin{aligned}&\alpha(x_t, z_t) \\ &= p(y_t|x_t, z_t)p(c_{1:t}) \int p(y_{1:t-1}, x_t, x_{t-1}|c_{1:t-1})p(z_t|c_t)\,\mathrm{d}x_{t-1} \\ &= p(y_t|x_t, z_t)p(c_{1:t}) \int p(y_{1:t-1}, x_t|x_{t-1}, c_{1:t-1})p(x_{t-1}|c_{1:t-1})p(z_t|c_t)\,\mathrm{d}x_{t-1} \\ &= p(y_t|x_t, z_t)p(c_{1:t}) \int p(y_{1:t-1}|x_t, x_{t-1}, c_{1:t-1})p(x_t|x_{t-1}, c_{1:t-1})p(x_{t-1}|c_{1:t-1})p(z_t|c_t)\,\mathrm{d}x_{t-1} \\ &= p(y_t|x_t, z_t)p(c_{1:t}) \int p(y_{1:t-1}, x_{t-1}|c_{1:t-1})p(x_t|x_{t-1})p(z_t|c_t)\,\mathrm{d}x_{t-1}\end{aligned}$$

As done previously in the standard state-observation model, we now define a normalized distribution $\hat{\alpha}(x_t, z_t) = p(x_t, z_t|y_{1:t}, c_{1:t})$ and introduce a similar scaling factor $s_t = p(y_t|y_{1:t-1}, c_{1:t})$. Using d-separation in the graph, we can then derive a recursive equation for $\hat{\alpha}(x_t, z_t)$:

$$\begin{aligned}s_t\,\hat{\alpha}(x_t, z_t) &= p(y_t|y_{1:t-1}, c_{1:t})\frac{\alpha(x_t, z_t)}{p(y_{1:t}, c_{1:t})} \\ &= p(y_t|y_{1:t-1}, c_{1:t})\frac{\alpha(x_t, z_t)}{p(y_t|y_{1:t-1}, c_{1:t})p(y_{1:t-1}, c_{1:t})} \\ &= p(y_t|y_{1:t-1}, c_{1:t})\frac{\alpha(x_t, z_t)}{p(y_t|y_{1:t-1}, c_{1:t})p(y_{1:t-1}|c_{1:t})p(c_{1:t})} \\ &= \frac{\alpha(x_t, z_t)}{p(y_{1:t-1}|c_{1:t-1})p(c_{1:t})}\end{aligned}$$

where we exploited conditional independence in the graph to show that the observations up to time $t-1$ are independent of c_t. This gives us the final form for the normalized recursive equation:

$$s_t\,\hat{\alpha}(x_t, z_t) = p(y_t|x_t, z_t) \int p(x_{t-1}|y_{1:t-1}, c_{1:t-1})p(x_t|x_{t-1})p(z_t|c_t)\,\mathrm{d}x_{t-1} \tag{6.5}$$

where we used the product rule decomposition to derive $p(y_{1:t-1}, x_{t-1}|c_{1:t-1})$ $= p(x_{t-1}|y_{1:t-1}, c_{1:t-1})p(y_{1:t-1}|c_{1:t-1})$.

Once again, we can view this equation as the propagation of the prior state belief through a prediction step, subsequently followed by an update step in which the occurrence of the observation y_t is taken into account. In the prediction step, we see that our belief about x_t and z_t is expressed as the product of two distinct distributions. The first corresponds to the usual propagation of our normalized prior over x_{t-1} through the prediction model $p(x_t|x_{t-1})$. The second is the noise prediction distribution for the new input c_t provided by the RVM model. In other words, the predicted belief over x_t and z_t is the simple product of the distribution of each variable provided by the component of the model that does not take the observation y_t into account. Note that this is analogous to the prediction step in the standard state-observation model. Finally, the predicted belief over x_t and z_t can be used as a prior and the contrapositive of the observation distribution $p(y_t|x_t, z_t)$ can be evaluated, yielding the updated belief for the state and noise variables.

Note that for systems with assumed Gaussian noise, this last operation leads to the convenient closed form expressions for recursive filtering, thanks to the properties of linear Gaussian models (A.2). In our case however, the peculiar form of the observation distribution prevent us from pursuing a similar analysis. It is quite common, for complex Bayesian networks, that an exact analysis actually leads to intractable forms for the hidden variables posteriors. Thus, exact inference is impossible, and while the suggested analysis showed more consistency regarding the Bayesian framework, approximations still have to be done. Two approximate inference methods can be suggested here, namely the variational Bayes methods, and *expectation propagation* (Minka, 2013). Roughly speaking, both methods exploit a similar principle, and approximate the intractable distribution by a product of independent factors representing distinct subsets of hidden variables. These factors are represented by convenient functional forms (often chosen to be in the exponential family) and their parameters are found by minimizing the Kullback-Leibler divergence between the approximate distribution (*i.e* the product of factors) and the exact distribution over the hidden variables. One illustration of these methods, and a departing point for pursuing the analysis of inference in our model, can be found in (Sarkka and Nummenmaa, 2009). Note that these developments are kept for future work.

6.2.5 Inference in the discriminative model

We now address inference in the discriminatively trained model. As explained in chapter 3, this specific training procedure results in an equivalent discriminative model which for the analysis provided before does not hold. Because the causal interpretation of the model changed, the noise variable

z_t and the state x_t are no longer coupled through the generative observation distribution, and inference is greatly simplified. As done during training, we can marginalize out the hidden variables z_t and $\mathbf{w}$ required to evaluate the posterior distribution $p(x_t|x_{t-1}, y_t, c_t)$. Recall that this last expression is implicitly provided by the recursive filtering equation (3.1).

Unlike the previous generative case, we can not easily marginalize out the posterior distribution over $\mathbf{w}$ using standard RVM equations. This is because discriminative training did not involve the evaluation of the posterior distribution of the noise level $\mathbf{z}$ corresponding to the noise observed at each input sample within the training set. Note that for the generative case this was automatically done in the expectation step of the EM procedure. In our approach, one solution to this problem consists in sampling the noise level $\mathbf{z}$ from the model, based on parameters we found after training. This approach was previously adopted in (Goldberg et al., 1998) in the context of heteroscedastic regression with GPs. In the following discussion, we omit the marginalization over $\mathbf{w}$ for clarity in the equations. Note that marginalization of the posterior distribution over $\mathbf{w}$ is implicitly done when predicting the new noise level z_t.

Recall from (3.1) that inference now consists in evaluating the posterior $\hat{\alpha}(x_t) = p(x_t|y_{1:t})$ through the equation:

$$s_t\hat{\alpha}(x_t) = \int\int \mathcal{N}\big(y_t|g(x_t), exp(z_t)\big)p(z_t, \mathbf{z}|c_t, \mathcal{D}, \alpha, \sigma_\nu) \int \hat{\alpha}(x_{t-1})p(x_t|x_{t-1})\,\mathrm{d}x_{t-1}\,\mathrm{d}\mathbf{z}\,\mathrm{d}z_t \tag{6.6}$$

where $\mathcal{D}$ denotes the training set data and $s_t = p(y_t|y_{1:t-1})$ the scaling factor. The distribution over z_t and $\mathbf{z}$ can be decomposed as follow, without loss of generality:

$$p(z_t, \mathbf{z}|c_t, \chi, \alpha, \sigma_\nu) = p(z_t|c_t, \mathbf{z}, \alpha, \sigma_\nu)p(\mathbf{z}|\chi, \alpha, \sigma_\nu)$$

In this last expression, the first term on the right-hand side is given by the RVM prediction equation for a new input c_t. The evaluation of the second term is obtained through sampling, as described hereafter. In the following, and as suggested previously in the generative case, we approximate the integral $\mathbf{z}$ in (6.6) with a point estimate. Finding the exact distribution $(\mathbf{z}|\chi, \alpha, \sigma_\nu)$ is not necessary in our case, as a point estimate over $\mathbf{z}$ allows for the straightforward exploitation of standard RVM prediction equations.

Thus, we turn to a widely used sampling method known as *Gibbs sampling*. In a first step, we use the Bayes rule to write:

$$p(\mathbf{z}|\mathbf{x}, \mathbf{y}, \mathbf{c}, \alpha, \sigma_\nu) \propto p(\mathbf{x}|\mathbf{y}, \mathbf{z})p(\mathbf{z}|\mathbf{c}, \alpha, \sigma_\nu) \tag{6.7}$$

Following the standard Gibbs sampling procedure, we then sample by cycling through the different components of $\mathbf{z}$. Noting z_i the i^{th} component of $\mathbf{z}$, and $z_{\setminus i}$ the remaining components, we have:

$$p(z_i|z_{\setminus i}, \mathbf{x}, \mathbf{y}, \mathbf{c}, \alpha, \sigma_\nu) \propto p(x_{t=i}|y_{t=\{1:i\}}, z_i)p(z_i|z_{\setminus i}, \mathbf{c}, \alpha, \sigma_\nu) \tag{6.8}$$

where the first term of the right-hand side of (6.8) is given by the recursive filtering equation, and the last term is given by the RVM prediction equation for a new input c_t. At first, samples are drawn from $p(z_i|z_{\setminus i}, \mathbf{c}, \alpha, \sigma_\nu)$. Each sample is then rejected or accepted regarding the likelihood $p(x_{t=i}|y_{t=\{1:i\}}, z_i)$. Intuitively, this procedure corresponds to maximizing the *discriminative likelihood* of the noise level z_i.

Once a point estimate for $\mathbf{z}$ is found, it can be used straightforwardly within regular RVM prediction equation for z_t. Once again, the integral over z_t in (6.6) can be approximated through the methods suggested in the generative case.

Note that the suggested procedures for training and inference in the discriminative case have not been tested yet. More importantly, a sound analysis of the sparsification properties (following the original analysis of RVM presented in appendix C) would provide insights about the convergence behavior of the hyperparameter optimization procedure. In the worst case, *i.e* if training does not efficiently achieve sparsity, the resulting computational cost would be analogous to that of GPs.

6.3 Experiments

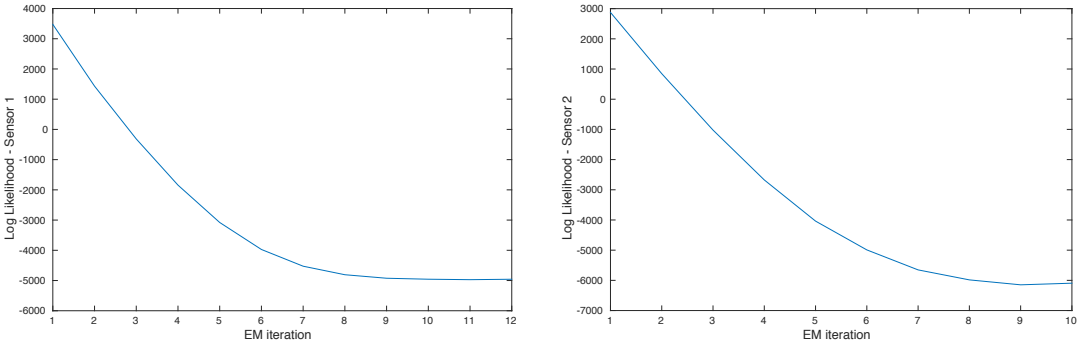
In the current state of our investigations, only the generative training and the fast approximate inference have been implemented and tested. Under this current implementation, we illustrate the performance of the model on real data, and subsequently combine the noise prediction model with the mRVM based selection component presented in chapter 5. Therefore we also illustrate the performance of the complete observation model suggested in chapter 3. Note that a comparison between generative and discriminative training, as well as the comparison between the two approximate inference methods in the generative case, are to be made in future investigations.

Fast approximate inference with generative training - Real data

In this experiment, and as done earlier, we consider the task of altitude estimation, and use the data gathered with our paparazzi quadrotor. Recall that altitude estimation was based on two exteroceptive sensors, *i.e* an ultrasonic sensor (that is referred to as sensor 1) and a barometer (referred to as sensor 2), both presenting distinctive behaviors. As discussed earlier in

this chapter, we ignore for now the correlation between the two sensor measurements and train two distinct RVM models that are subsequently used to predict the noise variance of each sensor. Here again, the RVM models exploit the Gaussian kernel with fixed input scale. The RVM input, *i.e* the contextual information c_t, is represented by a 3-dimensional vector including the two current measurements provided by the altitude sensors and the thrust command.

Since the hybrid type-2/EM training method is subject to oscillations, we stop the optimization process when we reach the log-likelihood local minimum. The log-likelihood progression of each model during training is shown Fig. 6.9, where we can see that both model converged in a few iterations, as typically observed with hard-EM algorithms.



(a) Log-likelihood progression the noise model of sensor 1.

(b) Log-likelihood progression for the noise model of sensor 2.

Figure 6.9: Log-likelihood progression during hybrid training for the two observation models.

After training, the noise models performance is then illustrated by evaluating the consistency of their prediction output on a validation set. The prediction mean and variance of the complete observation model ($p(y_t|x_t, c_t) \sim g(x_t) + \epsilon(c_t)$) of each sensor is shown Fig. 6.10 and Fig. 6.11. For both models, training results in highly sparsified data as, departing from a 5000 samples training set, the first model finally exploits 93 relevant vectors, and the second model only uses 20 relevant vectors. The difference in the number of relevant points is logically governed by the regularity of the altitude profile provided by each sensor. As seen in previous chapters, and here in Fig. 6.10, the sensor 1 (ultrasonic sensor) provides measurements at a higher frequency than for sensor 2, which is interpolated for synchronisation purpose. Furthermore, the altitude measurements provided by the sensor 1 presents numerous outliers, and these combined effects explain that the altitude profile of sensor 2 is usually smoother. Thus, it is logical that the noise model for the sensor 1 requires more relevance vectors, as the amplitude variations

in the noise are here significantly higher than for sensor 2.

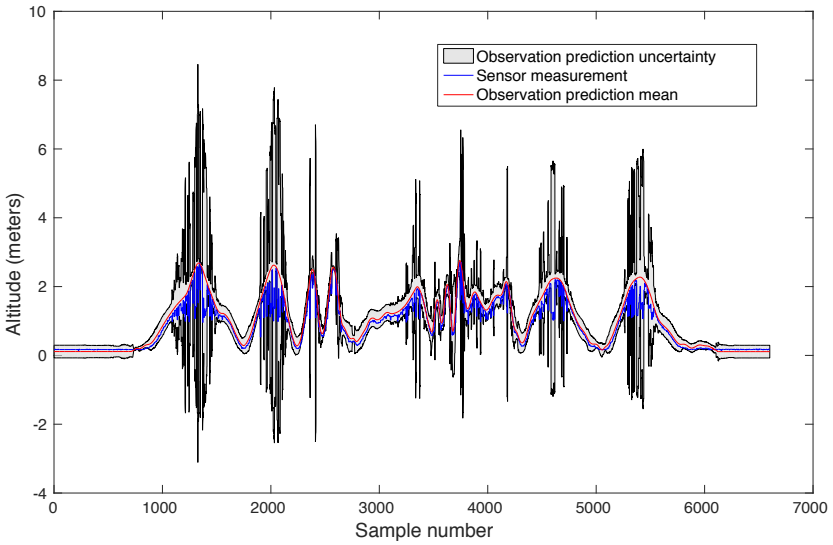


Figure 6.10: Observation predictions $p(y_t|x_t, c_t)$ for sensor 1 on the validation dataset. Boundaries illustrating the observation uncertainty corresponds to 3 standard deviations (3-sigma).

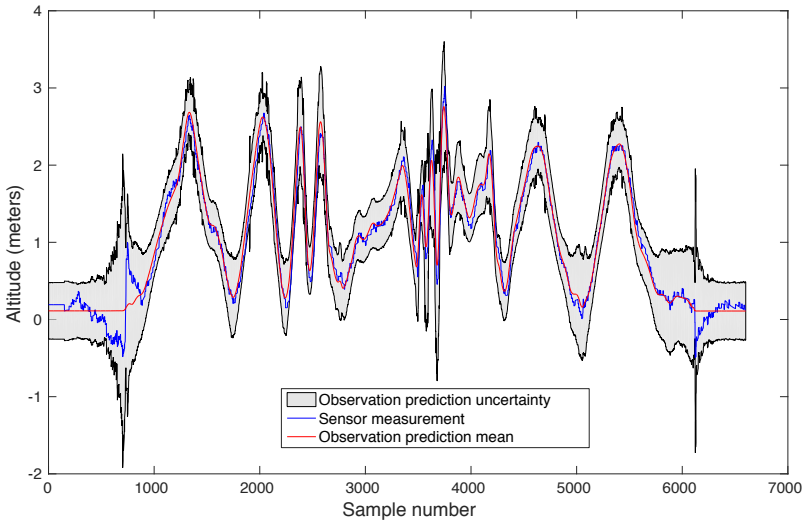


Figure 6.11: Observation predictions $p(y_t|x_t, c_t)$ for sensor 2 on the validation dataset. Boundaries illustrating the observation uncertainty corresponds to 3 standard deviations (3-sigma).

The accuracy of the observation noise on the validation set is more noticeable Fig. 6.12, where we magnified the Fig. 6.10 in the interval between sample 4200 and 5600. Similarly, the prediction output for sensor 2 is accurately consistent with the measurements, and we observe a tendency to overestimate the noise magnitude. In fact, while the barometer provides accurate measurements on the validation set, we can see Fig. 6.13 that the sensor measurements were less accurate on the training set. Recalling that the barometer is very sensitive to changes in air pressure and weather conditions, we understand that the predicted noise in the validation set is more coherent with the error seen in the training set.

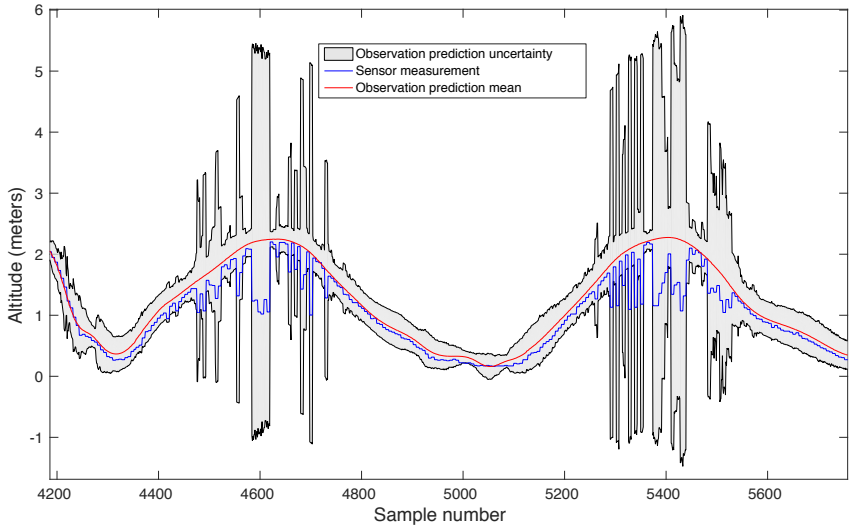


Figure 6.12: Zoom on the observation predictions $p(y_t|x_t, c_t)$ for sensor 1 on the validation dataset.

Combining noise adaptation and measurement selection

Both noise models are then used for altitude estimation, and combined with the measurement selection mechanism presented in chapter 5. Note that the context-dependent noise models can be straightforwardly exploited within a standard Bayes filter, following the fast approximate inference presented earlier. However, we aim here at illustrating the efficiency of the complete context-dependent observation model presented in chapter 3. In a first time, the two RVM noise models are trained following the type-2/EM training method. These two models are then used to create a set of 3 distinct filters, each of which using either the measurement provided by the sensor 1, the measurement provided by the sensor 2, or both measurements. We then train a mRVM model for selection following the approach suggested in chapter 5, and observe that all filters share a comparable activation rate in

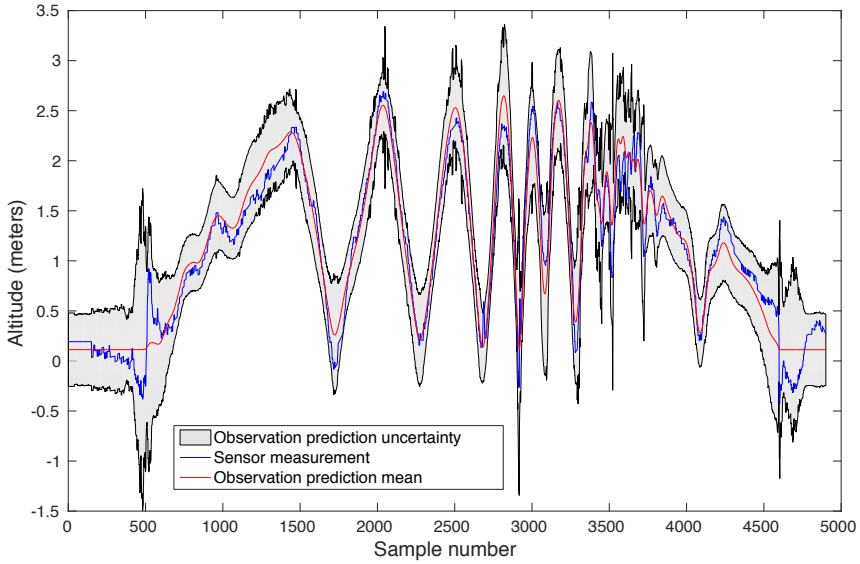


Figure 6.13: Observation predictions $p(y_t|x_t, c_t)$ for sensor 2 on the training dataset.

the training data. Thus, we do not remove any observation model from the bank. Note that both the noise prediction and the measurement selection components share the same contextual information.

The mRVM model is also based on a Gaussian kernel and exploits 94 relevance vectors for prediction. The average time required for selecting the active measurement subset, predicting the associated noise magnitude, and estimating the new state belief is of 0.002 seconds. Note that this is similar to the amount of time required in the experiment presented in chapter 5 where we did not use the noise adaptation component. In fact, the number of classes is here slightly lower (3 instead of 5 on the refined model) and the mRVM model is subsequently faster at making new predictions. However, this improvement is not really significant, but similar to the additional amount of time required here for predicting the noise magnitude, which is why the overall computation time is equal in both applications. Note that these values are obtained with non-optimized matlab code, and the approach is thus compliant with real applications constraints where we need to re-estimate the state belief at high frequencies.

Following the procedure for training and validation used in the previous chapters, we then illustrate the complete model performance on a previously unseen validation set. The resulting altitude estimation is shown Fig. 6.15. For comparison, Fig. 6.14 shows the altitude estimation previously obtained with the sole mRVM selection mechanism. As we can see, some outliers were not perfectly rejected but the noise prediction components locally help in

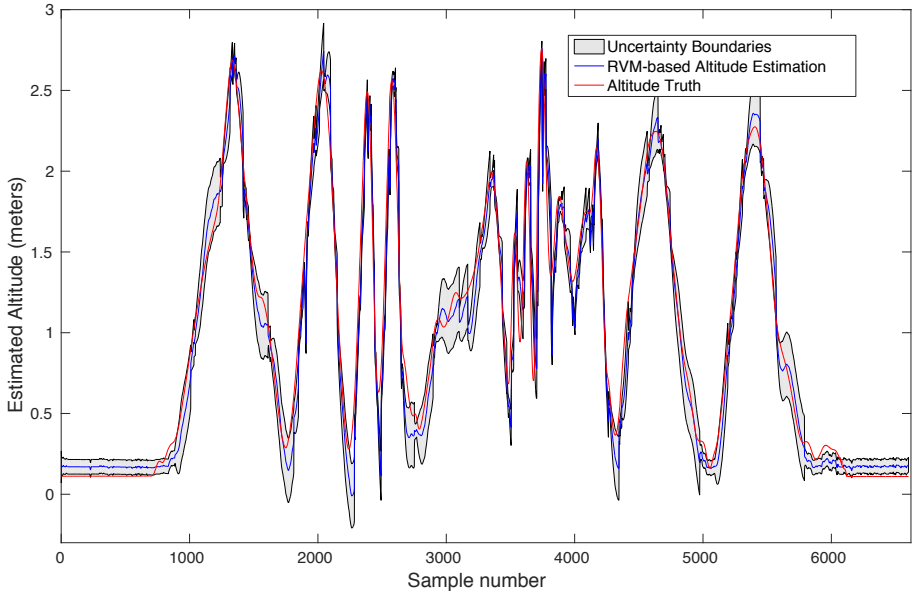


Figure 6.14: Altitude estimation with homoscedastic noise models and mRVM based measurement selection.

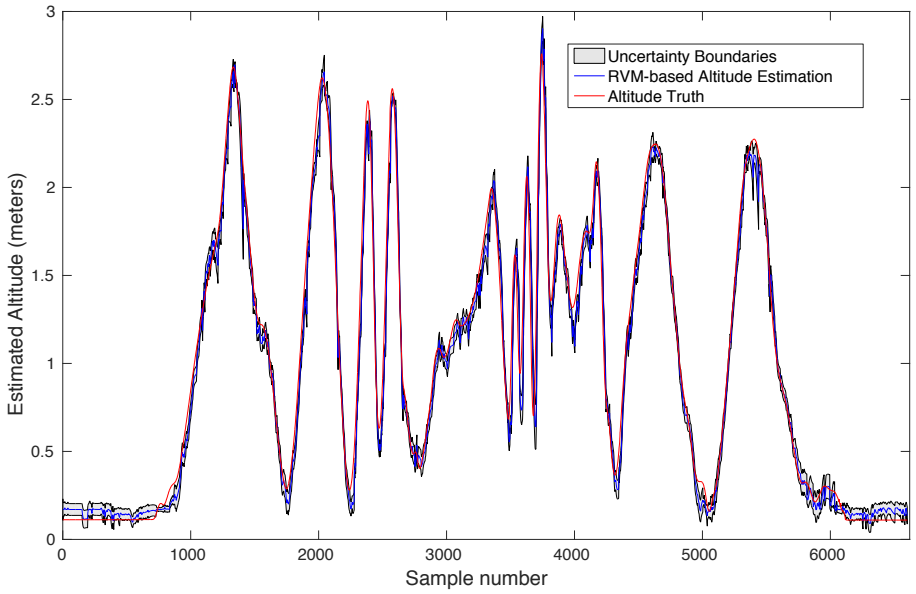


Figure 6.15: Altitude estimation with heteroscedastic noise models and mRVM based measurement selection.

preserving the state consistency. Unsurprisingly, they also help in improving the overall accuracy, and the complete model provides the best RMS error amongst all methods presented until now, with a score of 0.105. Recall that the mRVM model with homoscedastic noise models provided an RMS error of 0.152.

Clearly, we can notice that the resulting state uncertainty is significantly lower when we use the noise prediction models than with homoscedastic noise. While this result is obviously dependent on the homoscedastic noise magnitude that was manually tuned, we however observe that the introduction of heteroscedastic noise usually leads to a global deterioration of the state consistency. For example, the log likelihood score, *i.e*

$$\frac{1}{T}\sum_{t=1}^{T} log\big(p(x_t|y_{1:t}, c_{1:t})\big)$$

(where T denotes the number of samples in the training set) obtained with homoscedastic noise is of -0.57, while the complete model provides a score of -0.90. Thus, while they help in improving the general accuracy of the state estimate, the noise model obtained with generative training tend to underestimate the real noise in the system. Note that this behavior was predictable since the noise components were trained generatively, and the resulting state consistency was not considered during training. Clearly, this results pleads for the application of discriminative training when learning noise models. Furthermore, this also shows that the training method (or equivalently the modeling approach) is almost as much important as the model complexity. For example, we may already improve drastically the log likelihood score by exploiting discriminatively trained homoscedastic noise models.

6.4 Conclusion

6.4.1 Summary

In this chapter we have presented an extension of the state-observation model allowing for context-dependent adaptation of the observation noise. This extension relies on the RVM regression model that was presented earlier in chapter 5. When the model is trained in a generative manner, we saw that our approach could be mapped to the standard problem of heteroscedastic regression. However, the solution we provide slightly differs from pure regression as the noise-free component is not learned together with the noise distribution, but rather assumed to be known and defined by the physical characteristics of the sensor. This specific implementation heals the inability of heteroscedastic models in properly differentiating the

role that both noise-free and noise components should play in modeling the observation process, especially in the presence of strong outliers.

As stated in the previous chapter, the RVM sparsification property results from the type-2 maximum likelihood optimization, and we consequently design a dedicated type-2/EM hybrid training method preserving sparsity in the new model. The specific case of generative training yielding complex interdependence between the state variable and the noise magnitude, we present a fast approximate inference method. Alternatively, we derive new recursive equations that are more respectful of the dependencies encoded in the graph, while still benefiting from the sparification guarantees of the RVM model.

A theoretical analysis of inference and training in the discriminative case is concurrently discussed. This approach requires and alternative optimization technique for learning the hyperparameters of the RVM model, for which the RVM standard sparsification guarantees do not hold anymore. In the current state of our investigations, these later developments have not been tested yet and we restrict our experimental analysis on the generative case.

Experiments illustrate the accuracy of the noise prediction model as well as their generalization capability on new data. By combining the noise models with a measurement selection component, we also illustrate the performance of the complete context-dependent model presented in chapter 3. This model provides the best RMS error score obtained on our real data, and notably improves by 50% the error obtained earlier with standard rejection and homoscedastic noise. Unsurprisingly, we saw that if heteroscedastic noise models help in improving the state accuracy, generative training does not provide any improvement in terms of consistency. Thus, all models should be discriminatively trained for optimal performance.

6.4.2 Discussion

Inference approximations: benefits and drawbacks ?

While we can easily understand how ignoring some dependencies in the model modifies the inference equations, we however ignore how it can affect the general performance of the filter. As illustrated in the generative training case, the fast approximation method, and a more proper treatment, bring different benefits. In the first approach, we prevent the filter from basing inference on self-consistency, which was one of our original objectives. In the second approach, we saw that inference over the noise variables involves a pure prediction step with the RVM model, and a consistency 'correction' with respect to the generative observation model. Thus this latter approach might help in smoothing and tempering the errors produced by the sole RVM model.

In any case, further experimental investigations are required regarding proper Bayesian inference over the model as only performance evaluation and comparison between the two methods will provide significant insights. It is very likely however, that the most suitable inference method will depend on the specific application. Also, note that this reveals a new investigation direction. Indeed, if we primarily wanted to avoid relying only on self-consistency for adapting the model parameters, we can chose here to mix the information provided by an external component (the RVM noise prediction model) with the self-consistency constraints encoded in the complete model. Thus, we shall now investigate on the benefits of mitigating pure self-consistency with an external and independent model through different approximations.

Finally, recall that both approaches referred to as 'fast approximate' and 'proper Bayesian treatment' remain approximations in the Bayesian framework. Indeed, the weight parameter $\mathbf{w}$ is still marginalized out for both training and inference, while we should actually see it as an additional hidden variable and run inference for (z, w, x) together. This would obviously require to exploit approximate methods such as the variational Bayes framework, or expectation propagation, and the resulting model would not inherit the sparsification properties of the standard RVM model. However, investigations should be made regarding the possibility of preserving sparsity with other approximations than the type-2 maximum likelihood.

Training and inference complexity in the discriminative case

While augmenting Bayes filters with time-varying noise model plays a central role in trying to compensate the optimistic assumptions usually made by the classical model, the training method might also have major consequences over the system performance. As such, discriminative training seems promising in that it requires to run the filter during optimization while generative training focuses on the underlying emission and prediction processes. Discriminative training nevertheless brings some particular issues, since at first, it does not allow to use classical training (and sparsification) methods for the RVM model, but also because the optimization process of $\mathcal{L}_{discr}$ is much more complex. Indeed, each term of the discriminative loss function is strongly related to the preceding one as a direct consequence of the recursive equations used for state estimation. Classical RVM models already require the optimization of a nonconvex function, and we still need to study the consequences of the additional complexity introduced along with this specific loss function.

Besides this particular issue, the use of discriminative training also requires some additional approximation methods for inference. Experiments will provide a better insight on how expected benefits and inherent drawbacks of the discriminative method impact the system performance, by com-

parison to the much simpler generative approach.

Learning the length scale parameter

We observed that learning the noise model can easily lead to overfitting, where variations in the noise magnitude fit precisely each sample of the training set. In practice, this behavior appears with low kernel length scale parameter (< 0.4 in our case) which intuitively governs the smoothness of the prediction output. Tuning the kernel length scale parameter by hand appeared to be more convenient as it consequently led to better generalization capabilities, while maximum likelihood optimization logically leads to low values for the length scale, and consequently to overfitting.

If the effects of overfitting in the noise model are usually softened in the estimation process, this however shows that maximum likelihood is not really adapted in our application, especially if we use point estimates and not a proper prior distribution over the length scale parameter. It is then interesting to investigate on the generalization improvements brought by the use of a proper prior distribution, recalling that this can be equivalently seen as introducing regularization terms over the maximum likelihood score.

Current state of investigations

This chapter was principally dedicated to presenting the theoretical foundations for generative and discriminative modeling of context-dependent noise models. Experiments conducted so far concerned the simple task of altitude estimation for an UAV, and only a fast approximate method have been tested. Thus, further experimental investigations are required for comparing the different training and inference methods suggested. Moreover, the approach still has to be tested on more complex scenarios involving numerous sensors and a broad range of contexts.

6.4.3 Future directions

In the presented approach, the specific choice of the RVM model to provide the noise magnitude yields some additional approximations. This is because there is no closed-form expression when we marginalize out the Gaussian distribution over z_t, recalling that the noise model takes the form $\epsilon(c_t) \sim \mathcal{N}\big(0, exp(z_t)\big)$. For this reason, we suggested to approximate the marginalization through a point estimate or via the unscented transform, both resulting in a Gaussian distribution that is necessary for recursive filtering[4]. A more conventional approach for modeling the variance distribution of the observation model is to exploit the Gamma distribution that is

[4]Recall that the Kalman filter equations represent the state belief through a Gaussian distribution at each iteration thanks to the conjugate properties of the linear Gaussian model

the conjugate prior distribution of the inverse variance (precision) of a Gaussian. However, the sparse Bayesian technique can not be straightforwardly applied for modeling outputs whose probability mass take the form of a Gamma distribution. One solution would consist here in using two RVM models for predicting the two parameters that govern the Gamma distribution, hence increasing the computational cost of the model. Furthermore, marginalizing out the precision variable whose prior is defined as a Gamma distribution does not result in a new Gaussian but in a Student-t distribution. Thus approximations are still to be made in order to provide recursive filtering solutions. The point estimate method we used in our experiments being a rough approximation, we shall however test the improvements provided by the introduction of a Gamma prior, and, first of all, by using the efficient unscented transform.

One central aspect in all the approaches for training and running inference suggested in this chapter is the sparsification property of the standard RVM model. As discussed in the experiment section, this results in a a highly time-efficient model that can straightforwardly be exploited online. Generally speaking, it is now accepted that the ability of finding the relevant examples during training is a central question in any memory based approach, and we especially believe that this natural capability provided by the RVM model is of great importance. In our developments, we tried to preserve the conditions under which the RVM yields automatic relevance determination, and accordingly adapted our utilization of the global model. However, we could follow an opposite approach and straightly build new complex models which for we could ensure that the sparsification properties hold. The existence of a general set of conditions that one would have to respect in order to yield sparsification is however, and to our actual knowledge, neither known nor proved.

Chapter 7

Conclusion and future research

7.1 Summary of contributions

This thesis explored a new modeling paradigm for improving the standard Bayes filter performance both in terms of global accuracy and robustness towards erroneous measurements. Our theoretical analysis is conducted within the Bayesian framework, and eased with the probabilistic graphical model representation presented in chapter 2. The proposed improvements are driven by two identified causes of failure arising from the standard state-observation implementation. The first cause is that many optimistic assumptions are made when building the model: oversimplification of the real interdependencies in the system through the HMM representation, noise homoscedasticity, and a general tendency to represent too many unknown phenomenon within the noise components. The second cause is strongly related to the presence of unmodeled aspects in the model, and is a general modeling paradigm that we earlier referred to as the 'closed-world' assumption. Following this approach, assessments can be made about unmodeled or mismodeled phenomenons, based on the sole consistency of observations with the model. This paradigm constitutes an issue as we remain concretely blind to the actual phenomenon involved in the observation process, and the lack of information may mislead our decision regarding the proper manner of processing a new observation. These aspects are discussed in the introduction of the manuscript.

In an attempt to fix these issues, we investigate on enhancing the model expressiveness and exploiting additional information for explicitly modeling the complex phenomenon involved in the observation process. In this context, we believe that machine learning techniques, and especially non-parametric models, provide the most efficient tool for acquiring and encoding knowledge about the measurement generation process, based on a finite set

of training examples. However, the need for high computational efficiency was also of major importance in our work, since our original research context, *i.e* autonomous robotics, requires fast state estimation methods. This pleads for the exploitation of dedicated techniques like sparse Bayesian models that provide fast inference properties, but also for the development of specific approximations when running inference in the model. Similarly, we kept the HMM structure as the backbone of our work, as it leads to the highly efficient recursive filtering equations.

We begin by suggesting a new modeling paradigm for learning a robust observation distribution. The idea is to explicitly represent the presence of alterations in the observations via a measurement selection mechanism, and furthermore to allow for time-varying noise. This results in a state-observation model with two additional components taking as input some external data that is referred to as the contextual information. The components achieving measurement selection and noise adaptation are built as classification and regression machines. Regarding the training of the complete model, we investigate on an alternative technique, *i.e* discriminative training, that is rarely employed in the domain of Bayes filtering as the state-observation model is fundamentally designed as a generative model. With discriminative training comes an interesting debate concerning the graph structure we eventually use, as we show that this specific approach implicitly builds an equivalent discriminative model. While both generative and discriminative modeling approaches have some relative merits, we however recall that in the context of Bayes filtering, we never exploit the generative capabilities of the underlying state-observation model. Thus, it seems reasonable to optimize the model with respect to its performance as a state estimator rather than its performance in generating the true observations. This is even more debatable knowing that the model itself is built upon the assumption that some aspects of the real system can be ignored and encompassed within a noise distribution. Furthermore, we see that the choice of using a generative or a discriminative model also affects the complexity of inference. These ideas are discussed in chapter 3.

Some specific implementations are presented in the following chapters, based on the mixture of experts model for measurement selection in a first time, and then by exploiting the RVM model for selection and adaptation. These developments, as well as the resulting observation model, are summarized in Fig. 7.1, and introduce the following specific contributions:

- Introduce contextual information within the estimation process for explicitly modeling the measurement alterations.
- Drop out the standard concept of outliers as measurements lying too far from the probability mass defined by the time-homogeneous observation distribution. We suggest instead to measure the utility of a

measurement as its propensity to improve the state estimate.

- Use adaptive filtering techniques for measurement selection.
- Exploit the equivalent discriminative graph structure resulting from discriminative training of a generative model.
- Exploit discriminative training in a time-heterogeneous state-observation model.
- Suggest approximations in learning and inference that enforce the sparsification properties provided by the RVM model.

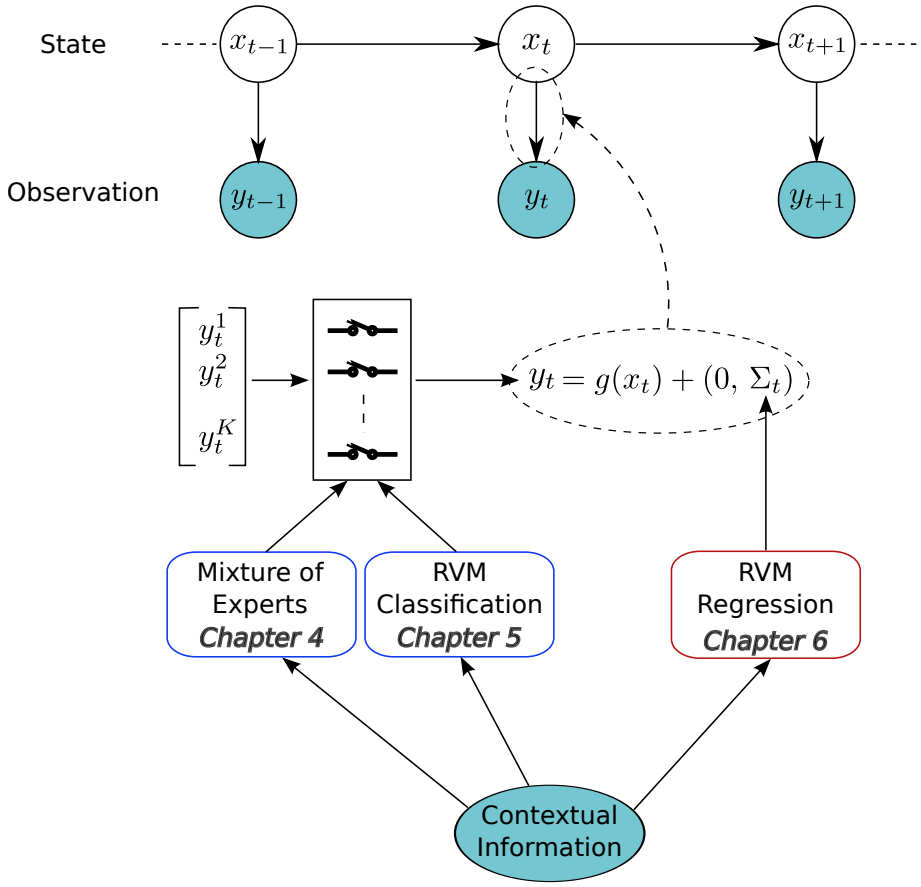


Figure 7.1: Global principle of the context-dependent state-observation model suggested in this work.

7.2 Future directions

7.2.1 Further investigations on the current approach

Investigating on the performance of generative vs. discriminative methods

From a practical point of view, discriminative training combined with both Mixture of Experts and mRVM for measurement selection has shown great robustness and consistency of their performance all along our experiments. Clearly, this showed us that optimizing the model parameters with respect to the filter estimate accuracy is a reliable approach. However, we have not explored and compared both discriminative and generative methods for each of our models: for this purpose, further experimental investigations are required. Note that, in a first time, discriminative training appeared to us as the most logical choice for training a measurement selection scheme, as we always considered that the utility of a measurement only depends on its ability to improve the state belief. Later in our investigations concerning the PGM framework, arose the debate between generative and discriminative training. While it has been shown in the case of Kalman filtering and for many HMM applications that discriminative training yields better results, this superiority is however not proved in the case of state-observation models with time-varying distributions. Moreover, some previous work (Ng and Jordan, 2001) has proved that the performance of generative and discriminative models depend on the amount of available training data.

Nonetheless, we have seen in this work that discriminative training may be interpreted as a graph remodeling whose new causal dependencies also influence the inference task. In our work, this is equivalent to turning the state-observation model into a simple state predictor using the observation y_t as input. Consequently, we could make direct use of the mixture of experts or direct measurement selection without changing the inference equations for each underlying filter. On the opposite, we saw that for the generative training of context-dependent noise models, exact inference does not lead to the standard recursive equations. Thus, great care has to be taken when choosing a training method, and the consequences regarding inference difficulty, as well as the necessary approximations, have to be examined.

Extension to multi-modal state belief and stochastic approximations

All the developments presented in this thesis are built upon the assumption that the state belief is unimodal and represented by a Gaussian distribution. Thus, we did not examine two common instantiations of Bayesian filtering that are multi-target tracking and particle filtering. Particle filtering automatically allows for the representation of multi-modal (or multi-

target) state distribution, and provide a solution for recursive filtering in non-linear and non-Gaussian state-space models. With the approximation introduced through the particle representation come important questions regarding convergence, *i.e*, for an infinite amount of particles, does the particle filter converges to the optimal filter, and does the resulting set of particles converge to the true state distribution? If some convergence proofs exist for the standard particle filtering instantiations (Crisan and Doucet, 2002), we can not guarantee that they still hold with our specific observation models, especially when we exploit the bank of filter approach. Thus, further investigations regarding the convergence properties should be done if we want to extend our model to particle filtering.

If we want to address applications requiring a multi-modal state distribution without exploiting stochastic approximation, one solution consists in representing the state distribution through a Gaussian mixture model (GMM). By assuming that the multi-modal state can be well described by a weighted sum of independent Gaussian distributions, we can guarantee that each state posterior can be used in the next iteration of recursive equations. More precisely, if we adopt a strict selection scheme as done in chapter 5, then we select the mixture of Gaussian provided by the most relevant filter, and straightforwardly use it for the next iteration of recursive filtering. If we prefer to mix the different filter outputs as done in chapter 4, then variants of the GPB approximation exist that allow for propagating a mixture of Gaussian through the state-observation model (Dovera and Della Rossa, 2011; Zhang et al., 2013). Thus, assuming that the multi-modal state distribution can be represented by a GMM allows us to exploit our approach with small modifications.

Multi-kernel approaches

If our experimental analysis proved that the RVM model provide an efficient solution for efficiently learning continuous and discrete mappings from simulated and real data, the contextual information we exploited until now was relatively homogeneous and simple. It is clear however that further applications of our approach to real estimation problems may have to deal with complex and potentially highly heterogeneous data (binary indicators, multi-dimensional measures, difference in amplitude ranges). Also, recall that one of our main objectives behind the introduction of learning and memory based methods was to avoid manual tweaking. Thus, we aim at providing a generic method that would not require specific adaptation of the models that govern measurement selection and noise adaptation regarding the nature of the contextual information. In fact, this problem is probably one of the most important issue that we have to solve in order to ensure real genericness of our approach. Note that this topic is also prevalent in the machine learning community when it comes to processing large scale

information.

As discussed chapter 5, one interesting improvement of the kernel methods is the extension to the multi-kernel approach. The multi-kernel methods intuitively exploit a set of distinct base kernel functions instead of a unique kernel, and provide predictions based through a weighted sum of the different kernels. Furthermore, as seen in (Damoulas and Girolami, 2008), the sparsification properties of the RVM model can also yield to a reduced set of relevant kernel functions. For this reasons, we particularly believe that the multi-kernel method is a key component in ensuring genericness of our approach.

7.2.2 Long term evolution

Improving old models or establishing new ones?

Our general motivation behind this work was to analyse and reconsider Bayesian filtering, starting from its theoretical foundations, while it is very usual to take the Kalman equations, and more generally the recursive Bayesian filtering equations, for 'granted'. We saw that under the PGM framework, we could easily re-formalize Bayesian filtering and build on top of it for improving its robustness and its accuracy. From the beginning of this work, it was deliberately chosen to enhance the original state-observation model instead of building new models from scratch, with the idea to keep low computational cost and a simple representation of the real system. However, we also support discriminative training which, as discussed chapter 3, implicitly yields a new graph structure.

Generally speaking, our theoretical analysis naturally leads to the idea that it is more relevant to formalize filtering from a discriminative point of view. This is because it is more reasonable to focus on encoding knowledge about the system state given the observations, than encoding knowledge over the observation generation process. Especially, and by definition, filtering means that we never exploit directly the generative capabilities of the state-observation model. Thus, filtering can be structurally seen as a discriminative problem, where our goal is to predict the state posterior given a state prior and an observation. In this formalism, we saw that the maximum entropy markov model naturally arises, and when transposed to undirected graphical models, we obtain a linear chain conditional random field. Both representations brings an interesting modeling property that would allow us to directly represent the probabilistic relationship between the state variable and both observation and context variables trough a unique component. In other words, we could replace our additional selection and adaptation models by a single edge $(c_t, y_t) - x_t$ in the graph. Following this approach, we believe that conditional random fields would provide an efficient solution, especially when combined with kernel methods (Lafferty et al., 2004).

However, if it is clear that generative modeling introduces additional misspecifications, we know that discriminative training helps in compensating for their effects on the global performance. Actually, recent investigations have shown that, in the debate about discriminative or generative models, one previously ignored answer yields the best performance: we shall *get the best of both worlds*[1]. Indeed, in (Bishop and Lasserre, 2007), the authors show that for a limited amount of training data, the best performance is provided by a mix of both approaches. This solution follows a quite neglected modeling principle in machine learning that is model averaging, and which explicitly recognizes that there is uncertainty in the likelihood function of any model. Thus, it is not sure that the future of filtering lies in pure discriminative modeling, but more likely in a combination of both generative and discriminative approaches.

[1]Excerpt from the title of (Bishop and Lasserre, 2007)

Appendix A

Mathematical properties

A.1 Properties of the linear Gaussian model

In a linear Gaussian model we are given a Gaussian distribution $p(x)$ and a Gaussian conditional distribution $p(y|x)$ whose mean is defined as a linear function of x so that:

$$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$$

$$p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})$$

where Λ^{-1} and L^{-1} are the precision matrices of each Gaussian, *i.e* the inverse of the covariance matrices.

Given $p(x)$ and $p(y|x)$, it can be showed (Bishop, 2006) that the joint distribution $p(x, y)$, the marginal $p(y)$ and the inverse conditional distribution $p(x|y)$ are all Gaussian. Moreover the marginal distribution of y and the conditional distribution of x given y are given by:

$$p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^t) \tag{A.1}$$

$$p(x|y) = \mathcal{N}(x|\Sigma[A^t L(y - b) + \Lambda\mu], \Sigma) \tag{A.2}$$

where we defined :

$$\Sigma = (\Lambda + A^t LA)^{-1}$$

If we consider $p(x)$ as a prior distribution of x, then $p(x|y)$ can be seen as the posterior distribution over x given the observation y. This conjugate property is a key ingredient in the derivation of the Kalman filter equations.

A.2 Matrix identities

In chapter 3 we make use of a basic identity which is:

$$(P^{-1} + B^t R^{-1} B)^{-1} B^t R^{-1} = PB^t (BPB^t + R)^{-1} \tag{A.3}$$

Similarly, we exploit the Woodbury identity that gives:

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1} \tag{A.4}$$

Appendix B

Inference in non-linear and non-Gaussian state observation models

B.1 Deterministic approximation

When prediction or observation involves non-linear functions but the assumption of Gaussian noise is still reasonable, we can exploit deterministic approximation methods. In this case we have a model of the form:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t|f(x_{t-1}), \Sigma_d)$$
$$p(y_t|x_t) = \mathcal{N}(y_t|g(x_t), \Sigma_o)$$

where either f, g or both are non-linear, and must be differentiable.

A first method consists in approximating the non-linear function f or g with a first-order Taylor series expansion evaluated at the maximum of the available prior distribution of the state. Following the common notation, we denote $x_{t|t}$ the posterior state belief so that $x_{t|t} \sim p(x_t|y_1, \ldots, y_t)$ and $x_{t|t-1}$ the prior state belief after propagation through the prediction model, *i.e* $x_{t|t-1} \sim \int p(x_{t-1}|y_1, \ldots, y_{t-1})p(x_t|x_{t-1})\,\mathrm{d}x_{t-1}$. This means that the function f is approximated by its first-order Taylor series approximation evaluated at $f(x_{t-1|t-1})$ and g by its approximation evaluated at $g(x_{t|t-1})$. This results in recursive equations similar to the Kalman filter, except that the Jacobian matrices of f and g replace the conventional observation and prediction matrices A and C. This method is known as the *extended Kalman filter* (EKF), and has proven to be unreliable for systems that are not close to linear in the time scale of the update intervals (Julier and Uhlmann, 1997).

A second approach exploits a different approximation paradigm: instead of approximating the non-linear function itself before propagating a Gaus-

sian distribution through it, it is better, and much simpler, to approximate the distribution resulting from the propagation through the non-linear component. The approximation is done by deterministically choosing a set of weighted samples called *sigma points* that optimally capture the properties of the distribution before passing through the non-linear function. The parameters of the resulting approximate Gaussian are subsequently evaluated using the propagated samples. This family of methods is referred to as *Sigma point Kalman filtering* (SPKF) (Merwe and Wan, 2003), and as such may be compared to the stochastic approximation method in which we try to find the best proposal distribution. However, while exploiting a similar concept, the SPKF relies on a deterministic sampling scheme. One popular deterministic transformation for computing the weighted samples is the *Unscented transform* (Julier and Uhlmann, 1996), which gives rise to the*Uscented Kalman filter*. The UKF proved to overcome many of the approximations issues brought by the EKF, and gives more accurate results since the unscented transform better captures the higher order moments of the linear function than the Taylor series expansion (Julier and Uhlmann, 2004).

The unscented transform and the UKF algorithm
Before providing the unscented Kalman filtering equations, we shall describe the unscented transform. This transform is closely related to the Gaussian quadrature method which basically aims at approximating the integral of the form

$$\int_{-\infty}^{+\infty} W(x) f(x)\,\mathrm{d}x$$

where $W(x)$ is a nonnegative function, and a Gaussian in our case.

This is done by choosing m points $x_1, \ldots, x_m$ and associated weights $w_1, \ldots, w_m$ so that

$$\int_{-\infty}^{+\infty} W(x) f(x)\,\mathrm{d}x = \sum_{i=1}^{m} w_i\, f(x_i) \tag{B.1}$$

In the unscented transform the sample points x_i are parsimoniously chosen by exploiting the symmetry properties of the Gaussian distribution around its axes. The actual theoretical foundations behind this sample selection method is behind the scope of this manuscript, and we prefer to present the practical implementation of the basic UKF.

Following the basic propagation and observation update sequence arising from (3.1), the UKF simply consists in applying two consecutive unscented transforms. The first approximation is used to compute the posterior after propagation through the prediction model ($p(x_t|y_1, \ldots, y_{t-1})$) and a second one to evaluate the posterior $p(x_t|y_1, \ldots, y_t)$.

Prediction step: we form at first a set of $2n + 1$ sigma points:

$$\chi_{t-1}^{[0]} = \mu_{t-1}$$
$$\chi_{t-1}^{[i]} = \mu_{t-1} + \sqrt{n+\lambda}\,[\sqrt{V_{t-1}}]_i$$
$$\chi_{t-1}^{[i+n]} = \mu_{t-1} - \sqrt{n+\lambda}\,[\sqrt{V_{t-1}}]_i$$

with $i = 1, \ldots, n$ and $[.]_i$ denotes the ith column of the matrix. λ is a scaling parameter defined by $\lambda = \alpha^2\,(n+k) - n$ where α and k are free parameters that control the spread of the sigma points around the mean, and whose optimal value is problem dependent.
The sigma points are then propagated through the non-linear prediction function:

$$\hat{\chi}_t^{[i]} = f(\chi_{t-1}^{[i]}) \quad i = 0, \ldots, 2n$$

Based on the propagated sigma points we can compute the mean and variance of the approximate predicted state distribution:

$$\hat{\mu}_{t|t-1} = \sum_{i=1}^{2n} W_i^m\,\hat{\chi}_t^{[i]}$$

$$\hat{V}_{t|t-1} = \sum_{i=1}^{2n} W_i^c\,(\hat{\chi}_t^{[i]} - \mu_{t|t-1})(\hat{\chi}_t^{[i]} - \mu_{t|t-1})^t + \Sigma_t$$

where the weights W_i^m, W_i^c are given by:

$$W_0^m = \frac{\lambda}{n+\lambda}$$
$$W_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta$$
$$W_i^m = \frac{1}{2(n+\lambda)} \quad i = 1, \ldots, 2n$$
$$W_i^c = \frac{1}{2(n+\lambda)} \quad i = 1, \ldots, 2n$$

with β an additional parameter that can be optionally used to introduce prior information about the distribution over x_t. Once we computed this predicted distribution over x_t, we can now take into account the observation and evaluate the posterior distribution $p(x_t|y_1, \ldots, y_t)$.

Update step: we form again a set of $2n + 1$ sigma points:

$$\chi_{t|t-1}^{[0]} = \hat{\mu}_{t|t-1}$$

$$\chi_{t|t-1}^{[i]} = \hat{\mu}_{t|t-1} + \sqrt{n+\lambda}\,[\sqrt{\hat{V}_{t|t-1}}]_i$$

$$\chi_{t|t-1}^{[i+n]} = \hat{\mu}_{t|t-1} - \sqrt{n+\lambda}\,[\sqrt{\hat{V}_{t|t-1}}]_i$$

Similarly, the sigma points are propagated through the non-linear observation function:

$$y_t^{[i]} = h(\chi_{t|t-1}^{[i]}) \quad i = 0, \ldots, 2n$$

Using these sigma points, we compute the predicted mean of the observation m_t, the covariance of the measurement S_t and the cross-correlation C_t between the state and the observation:

$$m_t = \sum_{i=0}^{2n} W_i^m \, y_t^{[i]}$$

$$S_t = \sum_{i=1}^{2n} W_i^c \, (y_t^{[i]} - m_t)(y_t^{[i]} - m_t)^t + \Sigma_o$$

$$C_t = \sum_{i=1}^{2n} W_i^c \, (\chi_{t|t-1}^{[i]} - \hat{\mu}_{t|t-1})(y_t^{[i]} - m_t)^t$$

And finally the resulting approximate mean and covariance of the state can be computed using:

$$\hat{\mu}_t = \hat{\mu}_{t|t-1} + K_t(y_t - m_t)$$

$$\hat{V}_t = \hat{V}_{t|t-1} - K_t S_t K_t^t$$

With the Kalman gain matrix:

$$K_t = C_t S_t^{-1}$$

B.2 Stochastic approximation

A popular family of approximate inference methods in non-linear and non-Gaussian models is the particle-based approach. In these methods, the state is not directly represented by a parametric distribution, as the exploitation of complex distributions often result in an intractable recursion equation

(3.1). The state belief is instead approximated by a set of weighted samples or particles, $x_t = \{x_t^m\}_{m=1}^M$, that represent the complex posterior distribution $p(x_t|y_1, \ldots, y_t)$. In practice, the basic idea of recursively propagating our prior belief through the prediction model $p(x_t|x_{t-1})$ and subsequently updating this distribution with the observation $p(y_t|x_t)$ is replaced by two equivalent approximate steps. In a first step, new samples are drawn from the prediction distribution $p(x_t|x_{t-1}^m)_{m=1}^M$, which corresponds to multiple hard-assignments of the integral in (3.1). Each of these new samples are then weighted proportionally to their observation likelihood $p(y_t|x_t^m)_{m=1}^M$.

These methods are referred to as *particle filtering*, or *sequential Monte-Carlo* approaches. The main issue in approximating the state distribution through a set of samples relies in the evaluation of their weights. Practically, in the simplest form of the algorithm, most of the samples quickly become irrelevant due to a negligible (or numerically negligible) weight. This phenomenon is known as *degeneracy* and leads to inconsistent estimates after a few iterations. Thus, many algorithms have been developed to tackle this issue, based on a resampling strategy that statistically preserves the samples with the highest weights, while trying to keep a better diversity among the samples. These approaches are known as *bootstrap filters*, *survival of the fittest* or *sequential importance resampling*, and the development of new methods is still an active research topic. Alternatively, degeneracy can be tackled through the exploitation of a better *proposal* distribution, in which the observation y_t is also taken into account instead of sampling from the unique prediction distribution.

When non-Gaussian distribution are involved, the sampling approximation showed better accuracy than the Gaussian approximation usually made in Bayes filters. Unfortunately, this approximation is also slower. For more details, a complete study of the particle filter variants can be found in (Doucet et al., 2001).

Appendix C

Sparse Bayesian learning analysis

The approach suggested by Tipping consists in rewriting the log likelihood in order to extract the term that depends on one particular hyperparameter α_i. As shown in (5.8), the complete analysis of the log likelihood solely rely on the term C that can be written:

$$\begin{aligned} C &= \sigma_r^2 I + \sum_{j=1}^{N+1} \alpha_j^{-1} \phi_j \phi_j^\intercal \\ &= \sigma_r^2 I + \sum_{j \neq i} \alpha_j^{-1} \phi_j \phi_j^\intercal \; + \; \alpha_i^{-1} \phi_i \phi_i^\intercal \\ &= C_{-i} \; + \; \alpha_i^{-1} \phi_i \phi_i^\intercal \end{aligned}$$

where ϕ_j denotes the j^{th} column vector of Φ and C_{-i} is the covariance matrix of the likelihood without considering the contribution of the vector ϕ_i. Note that this is equivalent to rewriting the likelihood function for $\alpha_i = \infty$.

Precisely, the log likelihood (5.8) depends on the determinant and the inverse of C that can now be written:

$$\begin{aligned} |C| &= |C_{-i}| \, |1 + \alpha_i^{-1} \phi_i^\intercal C_{-i}^{-1} \phi_i| \\ C^{-1} &= C_{-i}^{-1} - \frac{C_{-i}^{-1} \phi_i \phi_i^\intercal C_{-i}^{-1}}{\alpha_i + \phi_i^\intercal C_{-i}^{-1} \phi_i} \end{aligned} \tag{C.1}$$

where we made use of (A.4) and exploited the basic determinant properties.

Using these new expressions, we can express (5.8) as the sum of two terms:

$$\begin{aligned} \mathcal{L} &= ln\big(p(\mathbf{y}|\mathbf{x}, \alpha, \sigma_r)\big) \\ &= \mathcal{L}(\alpha_{-i}) + \mathcal{L}(\alpha_i) \end{aligned}$$

where $\mathcal{L}(\alpha_{-i})$ the log likelihood with the contribution of ϕ_i removed, and $\mathcal{L}(\alpha_i)$ the term relating the influence of ϕ, *i.e* depending on α_i. This last term is given by

$$\mathcal{L}(\alpha_i) = \frac{1}{2}\left(ln(\alpha_i) - ln(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i} \right)$$

where, for simplification, we defined the terms

$$s_i = \phi_i^\intercal C_{-i}^{-1} \phi_i$$

$$q_i = \phi_i^\intercal C_{-i}^{-1} \mathbf{y}$$

We can now compute the derivative of the log likelihood with respect to α_i, which is given by:

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \frac{\partial \mathcal{L}(\alpha_i)}{\partial \alpha_i} = \frac{\alpha_i^{-1} s_i^2 - (q_i^2 - s_i)}{2(\alpha_i + s_i)^2}$$

Setting this derivative to zero provides a stationary point given by:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i}$$

However, as a precision parameter, we recall that $\alpha_i > 0$, which requires $q_i^2 > s_i$. In the opposite case ($q_i^2 > s_i$), we see that the only solution corresponds to $\alpha_i = +\infty$. As can be seen in (C.1), the relative value of q_i^2 with respect to s_i may increase or decrease the value of the likelihood of the model. In practice, q_i^2 helps in increasing the likelihood, while s_i may reduce this effect, as a denominator of q_i^2. It is shown in (Faul and Tipping, 2001) that for $q_i^2 > s_i$, there is a single maximum for $\mathcal{L}$, while in the other case, the maximum is obtained at $\alpha_i \to \infty$. Thus, given the relative size of these factors, we can evaluate how a component ϕ_i helps in explaining the observed data, and if it can be pruned or not. This analysis leads to a faster training algorithm that fosters sparsification. More details about this method can be found in (Tipping et al., 2003).

Appendix D

French Summary

D.1 Introduction: Contexte de recherche

Ces travaux se focalisent sur une problématique fondamentale de la robotique autonome: l'estimation d'état. A l'heure actuelle, nous savons en effet que la plupart des approches permettant à un robot de réaliser une tâche quelconque requièrent tout d'abord l'extraction d'un **état** à partir de mesures capteurs bruitées. Ce vecteur d'état contient un ensemble de variables caractérisant le système à un instant t, comme la position du robot, sa vitesse, etc.

En robotique, et dans de nombreux autres domaines, le filtrage bayésien est devenu la solution la plus populaire pour estimer l'état d'un système de façon robuste et rapide. Cette technique repose grossièrement sur deux composantes principales. La première composante nous permet de prédire l'évolution de l'état dans le temps en se basant uniquement sur un modèle dynamique du système. La seconde composante permet, elle, de prendre en compte un ensemble de mesures capteurs afin de corriger l'état propagé 'à l'aveugle' à travers le modèle de prédiction.

En pratique, et particulièrement dans le cadre de la navigation autonome, la qualité des mesures capteurs est fortement dépendante du contexte de navigation. **Par contexte de navigation, nous entendons ici tout facteur physique, autre que les variables décrites par l'état, ayant une influence sur la qualité des mesures fournies par un capteur**. Par conséquent, la qualité des mesures capteurs peut s'étaler sur une échelle allant du fonctionnement nominal, en passant par un bruit de plus en plus important, jusqu'au moment ou la donnée n'est plus informative pour la tâche d'estimation. Ce phénomène est illustré Fig.D.1.

Dans la section suivante, nous détaillons plus précisément le modèle état observation qui repose derrière toute technique de filtrage bayésien. Dans une seconde partie nous examinerons plus en détail les avantages et défauts principaux de ce modèle. Ces sections, ainsi que le reste de ce résumé,

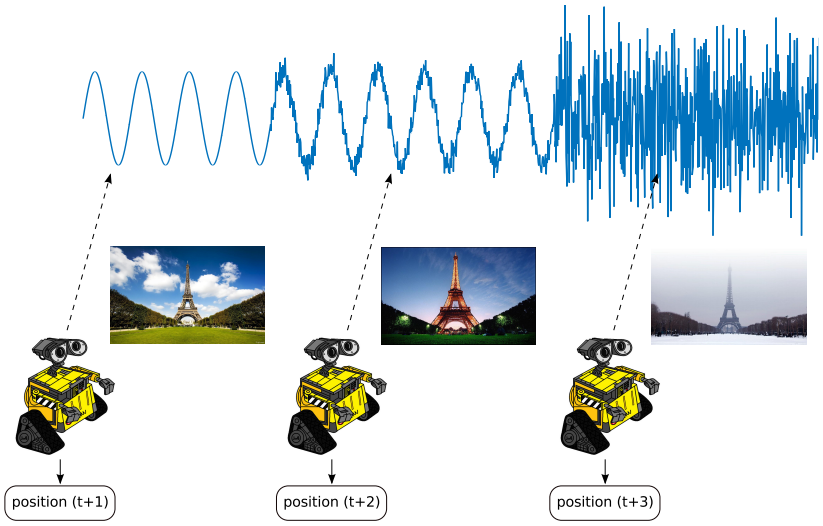


Figure D.1: Évolution des perturbations sur les mesures capteurs pour différents contextes de navigation.

exploitent le framework des modèles graphiques probabilistes, dont le lecteur trouvera une description détaillée dans (Bishop, 2006; Koller and Friedman, 2009).

D.2 Filtrage bayésien et modèle état observation

D.2.1 Le modèle état-observation

La première adaptation des modèles graphiques probabilistes aux systèmes dynamiques fut tout d'abord introduite dans (Dean and Kanazawa, 1989). Plus tard, l'emploi du terme *réseaux bayésiens dynamiques* fut proposé par les même auteurs. Les réseaux bayésiens dynamiques reposent, à l'origine, sur la propriété de Markov ainsi que sur l'hypothèse de distribution invariantes dans le temps.

Plus particulièrement, le modèle état observation repose originellement sur l'hypothèse de Markov d'ordre 1. L'évolution de l'état x_t dans le temps est donc décrite par une distribution stationnaire $P(x_t|x_{t-1})$. Pour chaque état x_t, les mesures perçues à l'instant t sont décrites par la distribution d'observation $P(y_t|x_t)$. Ces deux distributions se retrouvent respectivement sous forme de flèches horizontales et verticales dans le graphe représenté Fig.D.2.

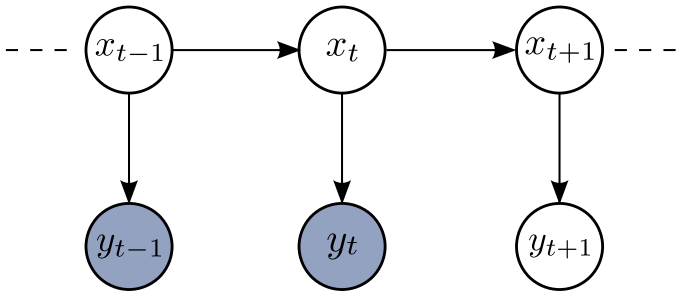


Figure D.2: Modèle état observation. Suivant le formalisme des modèles graphiques probabilistes, l'état x_t et l'observation associée y_t sont considérées comme des variables aléatoires représentées par des cercles. Les liens causaux entre variables sont représentés par des flèches représentant les distributions conditionnelles $P(x_t|x_{t-1})$ et $P(y_t|x_t)$. Les variables aléatoires grisées correspondent aux variables réellement observées (dans l'exemple, l'observation y_{t+1} n'a pas encore été perçue.)

L'exploitation de ce modèle, ou inférence, se divise traditionnellement en trois tâches distinctes:

- Le filtrage consiste à estimer la distribution de l'état x_t étant donné l'ensemble des observations perçues depuis le début de la séquence jusqu'à l'instant t.
- La prédiction consiste simplement à évaluer la distribution d'un état futur x_{t+n}, étant donné les observations $\{y_1, \ldots, y_t\}$.
- Le smoothing consiste enfin à évaluer la distribution d'un état x_k étant donné les observations $\{y_1, \ldots, y_T\}$ avec $k < T$.

Dans ce travail, nous nous focalisons uniquement sur la tâche de filtrage. Cette restriction a notamment des conséquences sur le type de modèles que nous allons définir, mais également sur la nature des fonctions de coût que nous utiliserons pendant l'apprentissage. Plus particulièrement, nous restreignons cette étude au cas ou les variables d'état et d'observation peuvent être représentées, ou approximées, par des distributions gaussiennes. Le cas de distributions non gaussiennes nécessitant l'exploitation de méthodes d'échantillonnage est à considérer comme une extension de ces travaux, et reste donc réservée pour de futurs développements.

Le filtrage bayésien est une méthode récursive qui consiste à évaluer la distribution de l'état normalisée $p(x_t|y_1, \ldots, y_t)$. Notons que cette distribution est dite normalisée car nous avons

$$p(x_t|y_1, ..., y_t) = \frac{\alpha(x_t)}{p(y_1, ..., y_t)}$$

En exploitant les indépendance conditionnelles dans le graphe, c'est-à-dire le principe de d-séparation (Koller and Friedman, 2009), il nous est possible de trouver une expression analytique pour évaluer la distribution normalisée de l'état. Cette expression correspond à la célèbre équation récursive de filtrage:

$$p(x_t|y_1,\dots,y_t) \propto p(y_t|x_t) \int p(x_{t-1}|y_1,\dots,y_{t-1}) p(x_t|x_{t-1}) \,\mathrm{d}x_{t-1}$$

. Pour des distributions gaussiennes, cette équation nous permet d'aboutir aux équations traditionnelles du filtre de Kalman cf.(Bishop, 2006).

De manière intuitive, cette équation décrit le déroulement récursif du processus d'estimation d'état qui repose sur deux étapes. Dans un premier temps, l'état obtenu à l'itération précédente est propagé à travers le modèle dynamique du système. Cette étape, dite de **prédiction**, correspond à l'intégrale $\int p(x_{t-1}|y_1,\dots,y_{t-1}) p(x_t|x_{t-1}) \,\mathrm{d}x_{t-1}$. Dans un second temps, la nouvelle distribution de l'état prédit (le résultat de l'intégrale) est pondérée par la probabilité de percevoir l'observation y_t dans ce nouvel état. C'est l'étape de correction, ou d'*update*.

Dans la partie suivante, nous abordons plus en détail la modélisation classique de la distribution d'observation, et les défauts principaux qui en découlent, et que nous essaierons de corriger.

D.2.2 Défauts du modèle état-observation

Suivant la méthode de modélisation classique, la définition de la distribution d'observation $p(y_t|x_t)$ repose sur deux éléments principaux. Dans un premier temps, le fonctionnement physique et supposé connu des capteurs est représenté par une composante déterministe qui associe à chaque état x_t un vecteur de mesures y_t. C'est la fonction d'observation que nous dénoterons ici h.

Afin de compenser pour les aspects méconnus ou simplement inconnus du processus de formation des mesures au niveau du capteur, une composante de bruit est ensuite ajoutée à la composante déterministe du modèle d'observation. Cette composante stochastique représente notre incertitude dans le modèle, que ce soit au niveau de la fonction d'observation, mais aussi globalement dans la structure très simplifiée du modèle état observation.

Ces deux composantes forment la distribution d'observation, comme représenté Fig.D.3.

Modèle d'observation simplifié

Cette modélisation canonique de la distribution d'observation présente un intérêt principal: sa simplicité. Cependant, elle introduit également trois problèmes majeurs que nous détaillons ci-dessous:

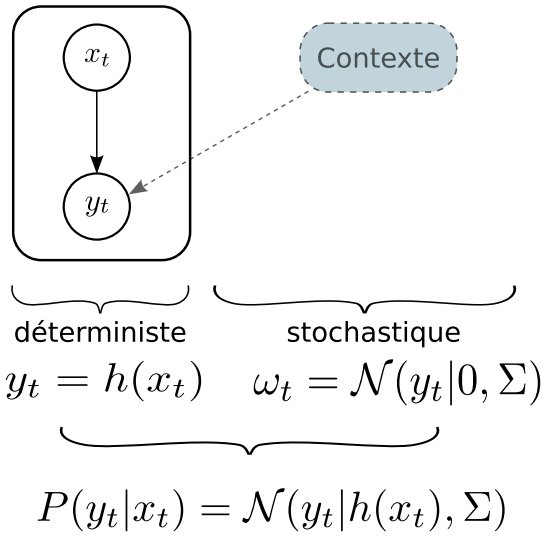


Figure D.3: Définition de la distribution d'observation au travers de deux composantes déterministes et stochastiques. L'exemple utilise un modèle de bruit gaussien centré en zéro tel que défini dans le cas du filtre de Kalman.

- **Modèle de bruit stationnaire:** historiquement, le modèle état observation a été bâti sur la théorie des chaînes de Markov, pour lesquelles l'hypothèse de distributions stationnaires simplifiait l'exploitation de ce type de modèle. En conséquence, les composantes de bruit du modèle état observation sont encore, le plus souvent, définis comme stationnaires. Comme nous l'avons vu, dans le cadre de la navigation autonome, l'existence de différents contextes de navigation modifiant la qualité des mesures est inévitable, et l'exploitation de distributions stationnaires devient problématique puisque nous sommes alors forcés de représenter plusieurs contextes (et amplitudes de bruits associés) sous un modèle unique.

- **Règles de réjection:** lorsque nous définissons une distribution d'observation stationnaire, nous introduisons implicitement l'existence d'outliers. Dans ce cas, un outlier est une mesure qui se situe trop loin de la masse centrale de la distribution d'observation, ou plus intuitivement, qui se situe hors de la zone d'incertitude définie par le modèle de bruit autour de la composante déterministe $y_t = h(x_t)$. Les outliers sont donc des mesures qui sont structurellement considérées comme des anomalies, et qui en pratique peuvent causer la divergence du filtre. En conséquence, il est souvent nécessaire d'adjoindre au modèle état observation une composante de réjection, afin de ne confronter le processus d'estimation qu'aux situations prévues par le modèle. En pratique,

cela revient à rejeter une mesure en se basant uniquement sur le fait qu'elle ne respecte pas le fonctionnement prévu par un modèle connu pour être erroné. Si la réjection de mesures fortement erronées reste utile, nous pensons néanmoins qu'il n'existe qu'un seul critère logique de réjection: la mesure permet elle d'améliorer ou non, la qualité de l'estimation d'état ?

- **Un modèle aveugle:** dans la continuité de la réflexion précédente, nous illustrons un dernier problème via la problématique de réjection. En l'état, lorsque nous décidons de rejeter une mesure, nous exploitons uniquement sa consistance avec le modèle, c'est à dire, sa conformité au comportement attendu du capteur tel que défini par le modèle d'observation. Si la mesure se situe hors de la zone d'incertitude du modèle, elle est donc rejetée. En utilisant, cette approche, nous venons donc détecter de manière indirecte l'existence d'un phénomène physique extérieur provoquant l'apparition d'une mesure erronée, sans l'observer. Cette exemple illustre le problème général du test de consistance, qui repose sur l'hypothèse forte que le modèle décrit intégralement tous les aspects du système nécessaires au processus d'estimation. Hors, ce modèle contient de multiples approximations, et par conséquent nous n'avons aucune garantie de fournir aux observations le traitement approprié. Une des conséquences de ce fonctionnement en aveugle est la tendance des filtres bayésiens à diverger.

Dans la section suivante, nous présentons une nouvelle approche de modélisation de la distribution d'observation qui vise à corriger ces problèmes, tout en limitant la complexité du modèle résultant.

D.3 Approche et principe d'implémentation

Le nouveau paradigme de modélisation de la distribution d'observation exploite la décomposition d'origine en une composante déterministe à laquelle est adjointe une composante de bruit représentant l'incertitude dans le modèle. Bien que fondamentalement très approximative, en comparaison des phénomènes physiques à l'origine de la formation des mesures au niveau capteur, cette modélisation présente deux intérêts principaux que nous souhaitons conserver: la simplicité, dont découle le faible coût en temps de calcul, ainsi qu'une grande robustesse.

Notre approche se différencie de la modélisation classique par l'introduction d'un modèle d'incertitude précis, qui évolue dans le temps afin de fournir la meilleure estimation possible étant donné l'ensemble de mesures à disposition. Plus précisément, notre solution repose sur trois idées fondamentales:

- La modélisation explicite de l'influence du contexte: pour ce faire, une nouvelle variable observée c_t représentant le contexte est introduite.

En fonction du contexte, la distribution d'observation, désormais hétérogène dans le temps, varie à la fois dans le sous ensemble de mesure qu'elle exploite à chaque instant t, ainsi que dans le bruit associé aux observations sélectionnées.

- L'exploitation des techniques d'apprentissage supervisé: la complexité de la nature du lien causal entre contexte et observation étant trop élevée, il nous est impossible de définir directement ce modèle. Par conséquent, la seule solution qui s'offre à nous consiste à apprendre ce modèle à partir de la donnée, suivant les techniques d'apprentissage supervisé classiques.
- La modélisation implicite de l'influence du contexte: cette notion plus subtile repose sur l'exploitation de méthode d'apprentissage alternatives, comme l'apprentissage discriminatif. Intuitivement, cette méthode consiste à optimiser les paramètres du modèle de manière à obtenir la meilleure estimation d'état, à l'inverse des méthodes classiques qui cherchent, elles, à expliquer les données d'entraînement selon les propriétés génératives du modèle. Il a été prouvé (Abbeel et al., 2005) que l'apprentissage discriminatif permet de compenser certaines imprécisions du modèle, comme dans notre cas l'existence de contextes pour lesquels la qualité des mesures varie.

D.3.1 Principes d'implémentation

En pratique, notre problématique s'apparente à la tâche de régression, et plus précisément de régression hétéroscédastique, qui vise à apprendre un modèle de régression pour un mapping $x_t, c_t \mapsto y_t$ tout en définissant une composante principale ainsi qu'un modèle de bruit tous deux variables dans le temps. Si en pratique cette approche a été suggérée et exploitée dans le cadre du filtrage bayésien (Ko and Fox, 2009), elle pose cependant un problème important dans le cas de données réelles et polluées, car il devient impossible de discerner quelle partie du signal doit être attribuée à la composante principale de celle qui doit être expliquée par le modèle de bruit lors de l'apprentissage. En cas de perturbations fortes sur les observations, cette approche se révèle donc inexploitable. Pour éviter ce travers, nous supposons ici que la composante principale (déterministe) du modèle d'observation est donc connue, ce qui en pratique revient à supposer que nous connaissons le fonctionnement physique de nos capteurs. Nous complétons ce modèle avec deux composants supplémentaires dont les rôles consistent respectivement à sélectionner un sous ensemble de mesures dans le vecteur d'observaiton, et à adapter le bruit d'observation des mesures sélectionnées. Ce principe est illustré Fig.D.4.

En pratique, nous pouvons représenter cette approche à travers la définition d'un nouveau modèle graphique générique, que nous représentons Fig.D.5.

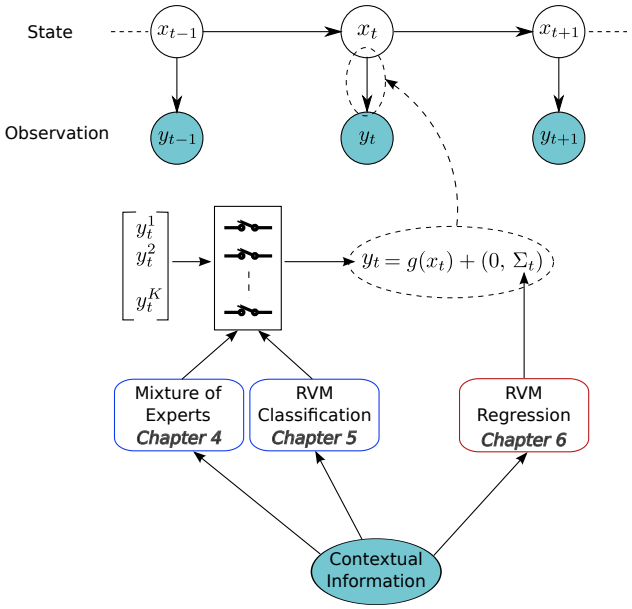


Figure D.4: Principe d'implémentation général, et détail des approches proposées dans ces travaux. La sélection par le mélange d'experts ainsi que l'adaptation du bruit via des modèles de type Relevance Vector Machine (RVM) seront détaillés par la suite. La sélection de mesure via RVM ne sera pas détaillée dans ce résumé, mais plus de détails sont donnés dans le chapitre 5 de cette thèse.

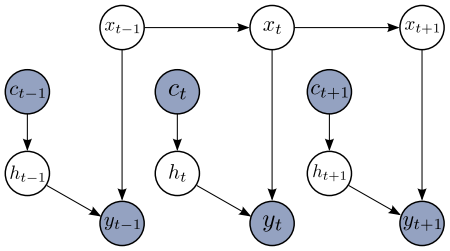


Figure D.5: Modèle graphique générique de l'approche proposée. c_t représente la variable de contexte observée. h_t représente une nouvelle variable cachée qui sera définie comme une variable discrète dans le cas de la classification, et continue dans le cas de l'adaptation du bruit.

Dans les sections suivantes, nous détaillons notre approche pour la sélection par le mélange d'experts ainsi que l'adaptation du bruit via des modèles de type Relevance Vector Machine. Ces travaux correspondent aux publications suivantes (Ravet et al., 2013; Ravet and Lacroix, 2014).

D.4 Illustration: sélection par le mélange d'experts

D.4.1 Introduction: modèle du mélange d'experts

Nous commençons cette section par introduire le modèle du mélange d'expert. Dans un second temps, nous verrons comment ce modèle peut être exploité pour la sélection de mesures.

L'approche du mélange d'experts repose sur une constatation simple: pour réaliser une tâche de régression ou de classification, il est souvent intéressant d'exploiter plusieurs modèles optimisés pour différentes régions de l'espace d'entrée, plutôt qu'un modèle unique. Suivant cette approche le mélange d'expert, ici détaillé pour la régression, requiert la définition d'une banque de modèle, ainsi que d'une fonction d'activation, communément appelée *gating network*. Ce principe est représenté Fig.D.6.

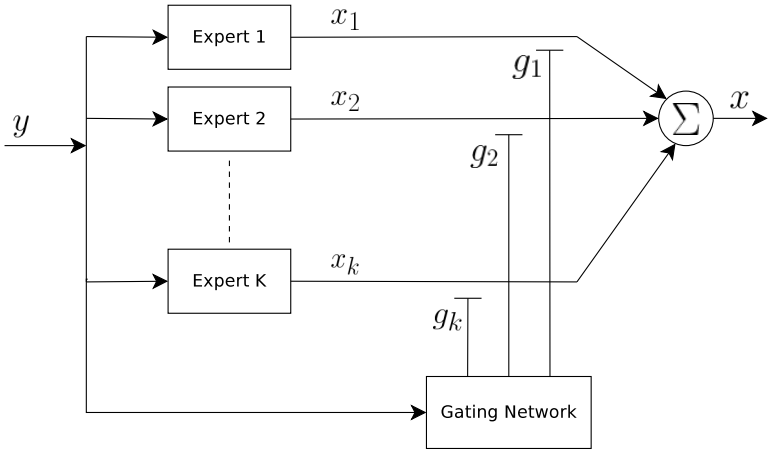


Figure D.6: Mélange d'expert pour la régression (Jacobs et al., 1991)

En pratique, chaque expert reçoit un vecteur d'entrée y et fournit une prédiction indépendante pour la sortie x. En parallèle, le gating network reçoit également le vecteur d'entrée y, et produit un ensemble de poids g_k attribués à chaque expert. Ces poids reflètent la confiance du gating network en la capacité de chaque expert à fournir la bonne variable de sortie x. La sortie globale du modèle est alors calculée simplement comme la somme pondérée de outputs de chaque expert. Traditionnellement, l'apprentissage du modèle permet de déterminer les paramètres optimaux de chaque modèle expert, ainsi que des régions d'activation (gating network).

D.4.2 Application à la sélection de mesures

En transposant l'approche du mélange d'experts à un mélange de filtres exploitant chacun un sous ensemble distinct des mesures dans le vecteur d'observation, nous pouvons donc automatiquement résoudre le problème de la sélection de mesures. La différence principale, pour notre application, réside dans le fait que chaque filtre (expert) exploite le vecteur d'observation en entrée, alors que le réseau de gating repose, lui, sur la variable de contexte. Cette approche est illustrée Fig.D.7.

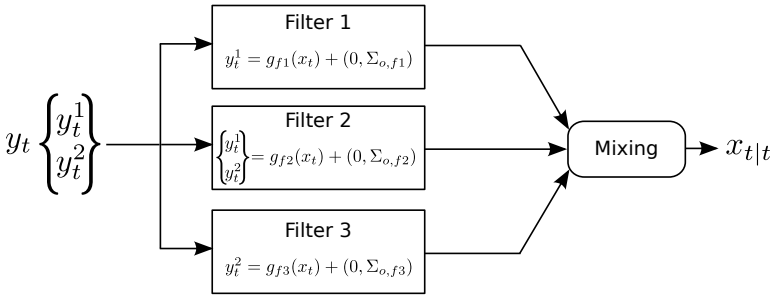


Figure D.7: Approche du mélange de filtres pour la sélection de mesures dépendant du contexte. L'illustration est donnée dans le cas d'un vecteur d'observation de dimension 2, qui résulte donc en l'exploitation de 3 filtres distincts exploitant respectivement les mesures y^1, (y^1, y^2) et y^2.

En pratique, l'étape d'apprentissage ne nécessite ici de définir que les paramètres du réseau de gating. Pour des raisons de simplicité, nous exploitons ici un modèle de région d'activation basé sur des noyaux gaussiens uni-modaux. Ce choix nous permet notamment d'exploiter l'algorithme *Expectation Maximisation* (EM) dont les détails sont fournis dans (Ravet et al., 2013).

Un aspect important de la méthode d'apprentissage réside dans la nature structurellement discriminative de ce modèle: pour optimiser les paramètres du modèle, dans le cas du mélange d'experts comme du mélange de filtres, nous allons chercher à expliquer au mieux les données d'apprentissage constituées de vecteurs d'observation et de l'état associé. Par conséquent, durant l'apprentissage, nous allons chercher à expliquer au mieux l'état x_t au travers des équations de filtrage, ce qui n'est pas le cas dans le cadre de l'apprentissage génératif traditionnel, qui cherche lui à optimiser les paramètres du modèle état observation de manière à expliquer au mieux l'évolution de l'état à travers la distribution de prédiction $p(x_t|x_{t-1})$ ainsi que les observations à travers la distribution d'observation $p(y_t|x_t)$. Notons que cela signifie donc que les équations de filtrage ne sont donc pas exploitées dans le cas de l'apprentissage génératif.

Cette spécificité a des conséquences importantes en terme de robustesse,

puisque nous prenons également en compte les erreurs de modélisation impactant la qualité de l'estimation. En pratique, nous avons notamment observé que la sortie du mélange de filtres restait toujours consistante, quelle que soit la qualité des paramètres de chaque filtre, ce qui n'est pas le cas pour un filtre traditionnel doté d'un mécanisme de réjection.

D.4.3 Expériences

Nous illustrons ici les résultats de l'approche du mélange de filtres dans cadre de l'estimation d'altitude d'un drone quadrotor équipé de capteurs fortement bruités. Les données d'entraînement sont obtenues à l'aide d'un système de motion capture qui fournit l'état du robot, et qui sont associées aux observation perçues par les capteurs. Ces données sont illustrées Fig.D.8.

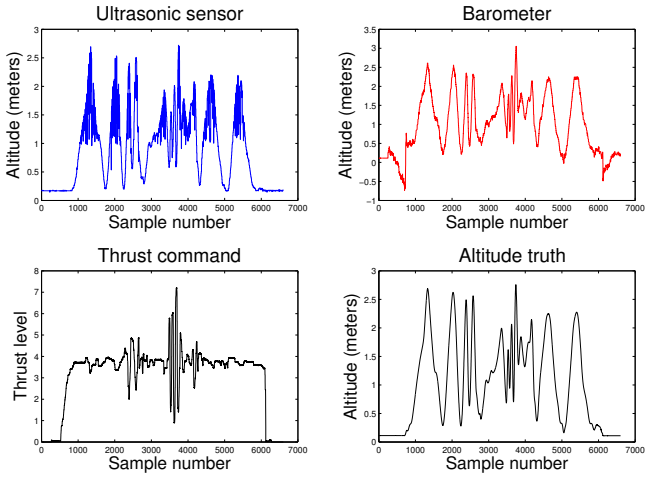


Figure D.8: Données d'entraînement comprenant (en haut) les mesures d'altitude fournies par un capteur à ultrason ainsi qu'un baromètre. La commande de poussée est également utilisée comme information de contexte (en bas à gauche), et la mesure d'altitude fournie par le système de motion capture nous fournit la vérité terrain.

Les informations de contexte sont ici définies par les mesures capteurs, ainsi que la commande en poussée. Ce choix se justifie par la présence de nombreux outliers dans les mesures ultrason, qui sont potentiellement provoqués par les perturbations électromagnétiques des moteurs. La commande en poussée, proportionnelle à la vitesse de rotation, nous semble donc pertinente dans le cadre de la sélection de mesures.

Les données d'entraînement contiennent environ 6000 échantillons, et l'optimisation des paramètres du gating network ne requiert que quelques minutes, correspondant à 50 itérations de l'algorithme EM. Après la phase

d'apprentissage, nous utilisons ce modèle sur des données de validation, qui diffèrent donc des données d'entraînement. La sortie du mélange de filtre est obtenue par la somme pondérée de l'estimation fournie par chaque filtre. Si nous dénotons μ_k et σ_k la moyenne et variance en sortie de chaque filtre, la somme pondérée correspond donc à l'approximation de type *generalized pseudo bayes* d'ordre 1 (GPB1) qui permet de représenter le mélange des sorties de chaque filtre au travers d'une distribution unique suivant les équations de mélange définies dans (Bar-Shalom et al., 2002):

$$\mu_{mix} = \sum_{k=1}^{K} g_k \mu_k$$

$$\sigma_{mix} = \sum_{k=1}^{K} g_k [\sigma_k + (\mu_k - \mu_{mix})(\mu_k - \mu_{mix})^T]$$

Un exemple d'estimation en sortie du mélange de filtre est fourni Fig.D.9. Comme nous pouvons le voir, la plupart des outliers sont rejetés correctement, même si certaines mesures erronées polluent ponctuellement l'estimation. Deux causes principales peuvent expliquer ces erreurs: il est d'abord possible que ces mesures correspondent à des types de contextes absents dans les données d'entraînement. En conséquence, nous n'avons pas de garantie de les traiter correctement. Deuxièmement, il est aussi possible que le modèle de noyaux gaussiens uni-modaux exploités pour la définition des régions d'activation reste relativement trop simple par rapport à la complexité du problème de la sélection des mesures. Une complexification de ce modèle est donc à envisager.

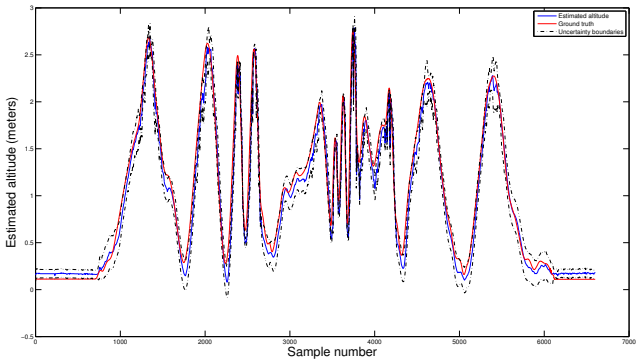


Figure D.9: Estimation d'altitude et incertitude associée en sortie du mélange de filtres sur des données de validation.

Dans la Fig.D.10, nous comparons l'erreur sur l'estimation d'altitude fournie par le mélange de filtres avec celle fournie par un filtre de Kalman

complété par une méthode de réjection de type 3-sigma. Clairement, le mélange de filtre fournit ici de bien meilleures performances que le filtre de Kalman, qui malgré une longue phase de réglage des paramètres, présente toujours des cas de divergence significatifs. De fait, la méthode de réjection se révèle très sensible au paramétrage des distributions de prédiction et d'observation du filtre, là ou le mélange de filtres présente une grande robustesse grâce à l'apprentissage discriminatif.

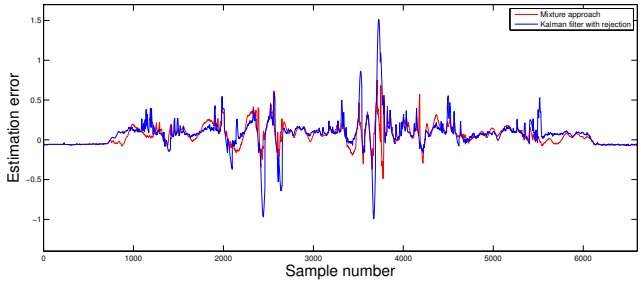


Figure D.10: Erreur sur l'estimation pour le mélange de filtres et un filtre de Kalman avec réjection basée 3-sigma.

D.4.4 Conclusion

Cette première approche pour la sélection de mesures nous fournit tout d'abord une preuve de l'applicabilité de techniques de sélection dépendant du contexte. Notamment, nous avons vu que le modèle proposé fournit de bonnes performances malgré un modèle des zones d'activation extrêmement simple. Ce modèle nous permet par ailleurs d'exploiter le mélange de filtres sans surcout significatif en temps de calcul.

Nous illustrons également qu'il n'est pas forcément nécessaire de définir la variable de contexte par le biais de nombreuses mesures additionnelles caractérisant l'environnement. A l'inverse, nous prouvons ici que l'ensemble des mesures contient un ensemble de patterns identifiables par des méthodes d'apprentissage. Ces patterns ne représentent pas directement le contexte, mais son incidence sur la formation des mesures au niveau capteur.

Comme nous l'avons vu, une piste d'amélioration majeure consiste à complexifier le modèle des zones d'activation du gating network. Pour ce faire, il est notamment intéressant d'explorer la piste des modèles de type non paramétriques. Cette piste a été explorée dans une publication additionnelle (Ravet et al., 2014).

D.5 Illustration: adaptation du bruit à l'aide de modèles non-paramétriques et sparses

D.5.1 Introduction: intérêt des modèles non-paramétriques

Dans l'approche précédente, nous avons fait l'hypothèse forte que les zones d'activation pouvaient être représentées par des distributions gaussiennes uni-modales. Cette hypothèse n'est pas sans conséquence, et il est clair qu'un modèle plus complexe serait plus adapté, et plus pertinent dans le cadre de la définition d'un modèle générique.

Notre but est ici de prédire à chaque instant t l'amplitude du bruit de chaque mesure contenue dans l'observation y_t. Clairement, nous n'avons aucune idée de la nature du lien causal entre la variable de contexte et le bruit de mesure. Pour éviter d'introduire une hypothèse forte sur la forme fonctionnelle de cette distribution, nous décidons ici d'exploiter les modèles non-paramétriques. Cette approche très puissante repose uniquement sur la similarité d'une nouvelle variable d'entrée avec les données d'entraînement pour prédire une nouvelle sortie.

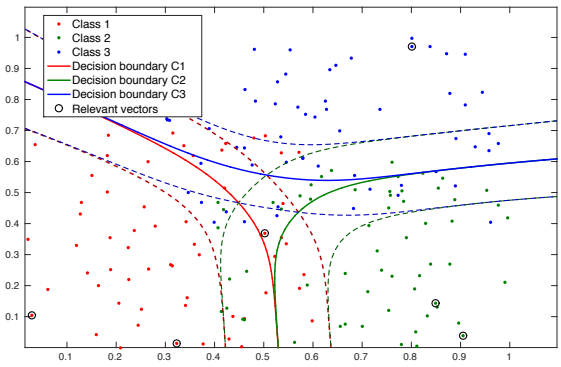


Figure D.11: RVM pour la tâche de classification. L'exemple fourni correspond à l'existence de 3 classes. Les points entourés en noir correspondent aux points d'intérêt (ou relevance vectors) conservés pour la prédiction, les autres sont rejetés.

Le problème principal de cette approche réside dans le temps de calcul conséquent de chaque nouvelle prédiction qui doit prendre en compte l'ensemble des échantillons des données d'entraînement. Par conséquent, dans le but de conserver un temps d'exécution permettant l'exploitation en temps réel, nous choisissons ici d'exploiter des modèles non-paramétriques dits *sparses*, qui nous permettent d'exploiter uniquement un sous-ensemble des échantillons des données d'entraînement. Le modèle RVM est particulièrement intéressant en ce sens puisqu'il permet de sparsifier la donnée

naturellement pendant la phase d'apprentissage. Le modèle RVM pour la classification est illustré Fig.D.11.

D.5.2 Adaptation du bruit

Le principe simplifié de notre approche est illustré Fig.D.12. Nous effectuons maintenant un court rappel des caractéristiques de modèle RVM ainsi que ses capacités de sparsification induites lors de l'apprentissage.

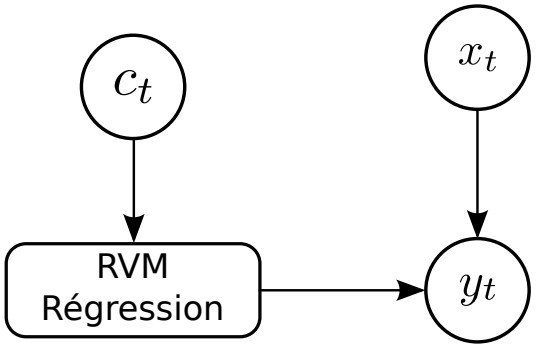


Figure D.12: Principe simplifié de l'adaptation du bruit à l'aide du modèle de régression RVM.

Relevance Vector Machine

Le modèle RVM fait partie de la famille des modèles linéaires étendus basés sur l'approche par kernel. En pratique, pour modéliser un mapping $x \mapsto y$ à travers une distribution de prédiction, cette technique revient à introduire la modélisation suivante:

$$p(y|x, w, \sigma) = \mathcal{N}(y \,|\, f(x), \sigma)$$

$$f(x) = \sum_{n=1}^{N} w_n \, K(x, x_n) + b$$

ou K représente la fonction kernel utilisée, qui intuitivement nous fournit la mesure de similarité entre une nouvelle entrée x avec les données d'entraînement x_n, et w_n le poids accordé à chaque échantillon x_n.

Sous deux condition spécifiques, la phase d'apprentissage conduit à la sparsification automatique des données d'entraînement (Tipping et al., 2003). La première condition est de définir une distribution à priori sur les poids w_n de type gaussienne centrée en zéro, tel que: $p(w_i|\alpha_i) \sim \mathcal{N}(w_i \,|\, 0, \alpha_i)$. La seconde condition est de marginaliser les variables w_n pendant l'apprentissage, ce qui correspond à la technique dite du maximum de vraisemblance de type

2 (Type-2 maximum likelihood). Par conséquent, la phase d'apprentissage se résume à l'optimisation suivante:

$$\underset{\sigma,\alpha}{\operatorname{argmax}} \left(p(y|x,\sigma,\alpha) = \int p(y|x,w,\sigma)p(w|\alpha)\,\mathrm{d}w \right)$$

Cette modélisation spécifique nous amène à construire le modèle graphique présenté Fig.D.13. L'apprentissage des paramètres du modèle de prédiction de l'amplitude des bruits, représentée ici par la variable z_t, requiert un traitement spécifique afin de conserver la conditions de sparsification du modèle RVM. En conséquence, en lieu de l'algorithme EM qu'il serait ici exact d'utiliser, nous introduisons une approche hybride EM-Type2 Maximum Likelihood qui nous permet d'optimiser les paramètres du modèle de façon itérative. Plus de détails concernant cette technique peuvent être trouvés dans (Ravet and Lacroix, 2014).

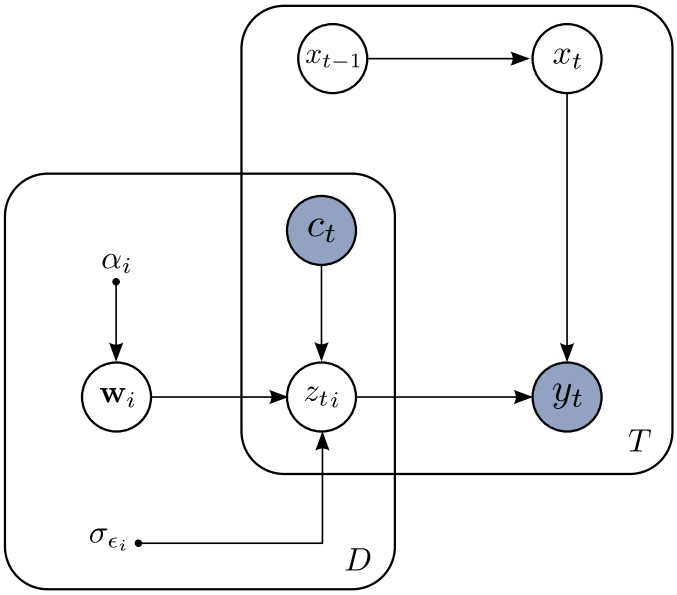


Figure D.13: Modèles graphique complet pour un modèle état observation dont les bruits de mesures varient en fonction du contexte, suivant le modèle RVM.

D.5.3 Expériences

Nous utilisons maintenant ce nouveau modèle dans le cadre expérimental introduit précédemment. Dans un premier temps, nous entraînons donc le modèle afin de prédire l'évolution du bruit d'observation des deux capteurs de mesure d'altitude. Un exemple de prédiction du bruit sur des données de validation est donné Fig.D.15.

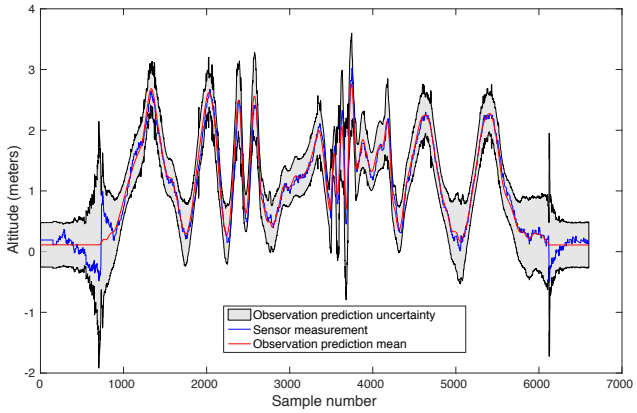


Figure D.14: Prédiction du bruit d'observation pour les mesures baromètre sur données de validation.

Le modèle de prédiction est ensuite utilisé dans les équations de filtrage. L'estimation d'altitude fournie par le système complet est illustré Fig.

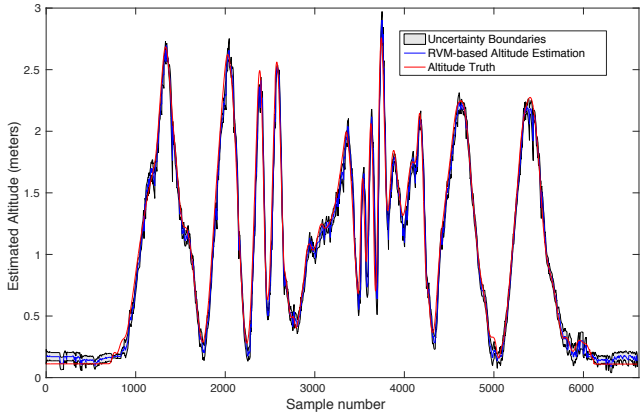


Figure D.15: Estimation d'altitude avec prédiction des bruits de mesures dépendants du contexte.

Comme nous pouvons le voir, l'exploitation de ce modèle résulte en une incertitude très faible en sortie du filtre. En pratique, ce modèle nous permet d'obtenir la meilleure performance en terme d'erreur sur l'estimation. Néanmoins, du fait de cette faible incertitude, les cas d'inconsistance avec la vérité terrain sont donc nombreux. Ce problème s'explique notamment par l'inexactitude de la méthode d'inférence que nous utilisons ici. Plus exactement, lorsque nous prédisons l'amplitude du bruit de mesure avant de l'exploiter au sein des équations de filtrage, nous ignorons l'existence d'une nouvelle dépendance entre la variable z_t et l'état x_t. Par conséquent,

les équations de filtrage ne sont plus directement exploitables et de nouvelles équations doivent être dérivées. Plus de détail concernant cette problématique peuvent être trouvés dans (Ravet and Lacroix, 2014).

D.6 Conclusion

Nous avons proposé deux approches pour la sélection et l'adaptation du bruit de mesures exploitables en temps réel. Ces techniques montrent des avantages significatifs en termes de performance en comparaison des approches traditionnelles, et présentent également l'avantage de ne pas nécessiter de temps de réglage des paramètres des modèles. Pour poursuivre ces développements, nous pensons que les modèles non-paramétriques, tels que le modèle RVM, sont d'un grand intérêt car capable de s'adapter facilement à toute application.

L'approche discriminative se révèle d'un grand intérêt dans la problématique de filtrage, puisque nous ne nous intéressons qu'à la performance finale du filtre, et non à la capacité du modèle à expliquer la donnée. Néanmoins, comme décrit dans (Minka, 2005), l'emploi de méthodes d'apprentissage discriminatives sur des modèles structurellement génératifs comme le modèle état observation n'est pas sans conséquence. En pratique, pour conserver une cohérence entre apprentissage et inférence, l'apprentissage discriminatif provoque l'apparition de nouveaux modèles, pour lesquels l'inférence change. Un grand soin doit donc être porté aux modèles graphiques équivalents, et aux dépendances qui sont souvent oubliées dans la littérature, ou les équations de filtrage ne sont pas remises en cause. Cette problématique nous montre que le formalisme des modèles graphiques probabilistes joue un grand rôle dans la compréhension et le développement des nouvelles techniques de filtrage.

Bibliography

Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y. Ng, and Sebastian Thrun. Discriminative training of kalman filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

Gabriel Agamennoni, Juan I Nieto, and Eduardo Mario Nebot. An outlier-robust kalman filter. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1551–1558. IEEE, 2011.

James H. Albert and Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

Armen Allahverdyan and Aram Galstyan. Comparative analysis of viterbi training and maximum likelihood estimation for hmms. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1674–1682. Curran Associates, Inc., 2011.

L. Bahl, P. Brown, P.V. de Souza, and R. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 49–52, Apr 1986.

Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

Yoshua Bengio and Paolo Frasconi. Input/output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7:1231–1249, 1996.

Christopher M. Bishop and Julia Lasserre. Generative or Discriminative? Getting the Best of Both Worlds. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 8*, pages 3–24. International Society for Bayesian Analysis, Oxford University Pres, 2007.

C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *Automatic Control, IEEE Transactions on*, 33(8):780 –783, aug 1988.

Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *In Proc. UAI*, pages 33–42, 1998.

P. Brisset, A. Drouin, M. Gorraz, P.S. Huard, and J. Tyler. The paparazzi solution. In *Micro Aerial Vehicles*, 2006. URL `http://paparazzi.enac.fr`.

K.P. Burnham and D.R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, 2002.

Wassim S. Chaer, Robert H. Bishop, and Joydeep Ghosh. A mixture-of-experts framework for adaptive Kalman filtering. *IEEE Transactions on Systems, Man, and Cybernetics*, 27, 1997.

W.S. Chaer, R.H. Bishop, and J. Ghosh. Hierarchical adaptive kalman filtering for interplanetary orbit determination. *Aerospace and Electronic Systems, IEEE Transactions on*, 34(3):883–896, Jul. 1998.

David Maxwell Chickering and David Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. In *Machine Learning*, pages 29–181, 1997.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

D. Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *Signal Processing, IEEE Transactions on*, 50(3):736–746, Mar 2002.

T. Damoulas, Yiming Ying, M.A. Girolami, and C. Campbell. Inferring sparse kernel combinations and relevance vectors: An application to subcellular localization of proteins. In *Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on*, pages 577–582, Dec 2008.

Theodoros Damoulas and Mark A. Girolami. Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.

Theodoros Damoulas and Mark A. Girolami. Combining feature spaces for classification. *Pattern Recogn.*, 42(11):2671–2683, November 2009.

N. de Freitas, R. Dearden, Frank Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a mars explorer. *Proceedings of the IEEE*, 92(3):455–468, 2004.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150, 1989.

Marc Peter Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic moment-based gaussian process filtering. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 225–232, New York, NY, USA, 2009. ACM.

Marc Peter Deisenroth, Ryan D. Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl Edward Rasmussen. Robust filtering and smoothing with gaussian processes. *CoRR*, abs/1203.4345, 2012.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer, 2001.

Laura Dovera and Ernesto Della Rossa. Multimodal ensemble kalman filtering using gaussian mixture models. *Computational Geosciences*, 15(2): 307–323, 2011.

Anita C. Faul and Michael E. Tipping. Analysis of sparse bayesian learning. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 383–389. MIT Press, 2001.

Nir Friedman and Daphne Koller. Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 50(1-2):95–125, 2003.

Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:963–996, 1998.

Mark Girolami and Simon Rogers. Hierarchic bayesian models for kernel learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 241–248, New York, NY, USA, 2005. ACM.

Paul W. Goldberg, Christopher K. I. Williams, and Christopher M. Bishop. Regression with input-dependent noise: A gaussian process treatment. In *In Advances in Neural Information Processing Systems 10*, pages 493–499. MIT Press, 1998.

N. J. Gordon and A. F. M. Smith. Approximate non-gaussian bayesian estimation and modal consistency. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(4):pp. 913–918, 1993. ISSN 00359246.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, March 1991.

A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. Mathematics in Science and Engineering. Elsevier Science, 1970.

Michael I. Jordan. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1993.

Michael I. Jordan and Lei Xu. Convergence results for the em approach to mixtures of experts architectures, 1993.

Biing-Hwang Juang and S. Katagiri. Discriminative learning for minimum error classification [pattern recognition]. *Signal Processing, IEEE Transactions on*, 40(12):3043–3054, Dec 1992.

Simon Julier and Jeffrey K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, 1996.

Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997.

S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar 2004. ISSN 0018-9219.

R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 393–400, New York, NY, USA, 2007. ACM.

Daniel Khashabi, Mojtaba Ziyadi, and Feng Liang. Heteroscedastic relevance vector machine. *CoRR*, abs/1301.2015, 2013.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5 (1):1–25, 1996.

Genshiro Kitagawa. Non-Gaussian State-Space Modeling of Nonstationary Time Series. *Journal of the American Statistical Association*, 82(400): 1032–1041, 1987. doi: 10.2307/2289375.

Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27 (1):75–90, 2009.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

John Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann, 2001.

John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 64–, New York, NY, USA, 2004. ACM.

Quoc V. Le and Alex J. Smola. Heteroscedastic gaussian process regression. In *In International Conference on Machine Learning ICML*, 2005.

James Loxam and Tom Drummond. Student-t mixture filter for robust, real-time visual tracking. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 372–385. Springer Berlin Heidelberg, 2008.

Miguel Lázaro-gredilla and Michalis K. Titsias. Variational heteroscedastic gaussian process regression. In *In 28th International Conference on Machine Learning (ICML-11*, pages 841–848. ACM, 2011.

D. J. C. Mackay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

David J.C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

David J.C. MacKay. Ensemble learning for hidden markov models. Technical report, 1997.

P.S. Maybeck. *Stochastic Models, Estimation and Control*. Number vol. 1 in Mathematics in science and engineering. Academic Press, 1982a.

P.S. Maybeck. *Stochastic Models, Estimation and Control*. Number vol. 2 in Mathematics in science and engineering. Academic Press, 1982b.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

Andrew McHutchon and Carl Edward Rasmussen. Gaussian process training with input noise. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 1341–1349, 2011.

R.K. Mehra. Approaches to adaptive filtering. *Automatic Control, IEEE Transactions on*, 17(5):693–698, Oct 1972.

Rudolph Van Der Merwe and Eric Wan. Sigma-point kalman filters for probabilistic inference in dynamic state-space models. In *In Proceedings of the Workshop on Advances in Machine Learning*, 2003.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. *CoRR*, abs/1301.2294, 2013.

Tom Minka. Discriminative models, not discriminative training. Technical Report MSR-TR-2005-144, Microsoft Research, October 2005.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, 2001.

J Pearl. Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, 29(3):241–288, September 1986.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

Joaquin Quiñonero-candela, Carl Edward Rasmussen, and Ralf Herbrich. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:2005, 2005.

Viswanath Ramamurti and Joydeep Ghosh. On the use of localized gating in mixture of experts networks, 1998.

Carl E. Rasmussen and Joaquin Quiñonero Candela. Healing the relevance vector machine through augmentation. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 689–696. ACM, 2005.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

A. Ravet, S. Lacroix, and G. Hattenberger. Augmenting bayes filters with the relevance vector machine for time-varying context-dependent observation distribution. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3039–3044, Sept 2014.

Alexandre Ravet and Simon Lacroix. Heterogeneous Bayes Filters with Sparse Bayesian Models: Application to state estimation in robotics. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD)*, page 10, Nancy, France, September 2014.

Alexandre Ravet, Simon Lacroix, Gautier Hattenberger, and Bertrand Vandeportaele. Learning to combine multi-sensor information for context dependent state estimation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5221–5226, 2013.

S. Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

S. Sarkka and A. Nummenmaa. Recursive noise adaptive kalman filtering by variational bayesian approximations. *Automatic Control, IEEE Transactions on*, 54(3):596–600, March 2009.

B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning. MIT Press, 2002.

R. Schubert and G. Wanielik. Unifying bayesian networks and imm filtering for improved multiple model estimation. In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 810 –817, july 2009.

G. Shafer and J. Pearl. *Readings in uncertain reasoning*. The Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1990. ISBN 9781558601253.

D. Sivia and J. Skilling. *Data Analysis: A Bayesian Tutorial*. Oxford science publications. OUP Oxford, 2006.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Arasanathan Thayananthan. *Template-based Pose Estimation and Tracking of 3D Hand Motion*. PhD thesis, Department of Engineering,University of Cambridge, 2005.

Arasanathan Thayananthan, Ramanan Navaratnam, Björn Stenger, Philip H. S. Torr, and Roberto Cipolla. Multivariate relevance vector machines

for tracking. In *ECCV (3)*, volume 3953 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2006.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005a. ISBN 9780262201629.

Sebastian Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20:2001, 2001.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005b.

Jo-Anne Ting, Evangelos Theodorou, and Stefan Schaal. Learning an outlier-robust kalman filter. In *Machine Learning: ECML 2007*, pages 748–756. Springer, 2007.

Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, September 2001.

Michael E. Tipping, Anita Faul, J J Thomson Avenue, and J J Thomson Avenue. Fast marginal likelihood maximisation for sparse bayesian models. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pages 3–6, 2003.

Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E. Hinton. SMEM Algorithm for Mixture Models. *Neural Comput.*, 12(9):2109–2128, September 2000.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IT-13(2):260–269, April 1967.

N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series with engineering applications*. Technology press books in science and engineering. Technology Press of the Massachusetts Institute of Technology, 1949.

David Wipf, Jason Palmer, and Bhaskar Rao. Perspectives on sparse bayesian learning. In *Advances in Neural Information Processing Systems 16*, page 2004. MIT Press, 2003.

Lei Xu, Michael I. Jordan, and Geoffrey E. Hinton. An alternative model for mixtures of experts. In *Advances in Neural Information Processing Systems*, 1995.

Chao Yuan and Claus Neubauer. Variational mixture of gaussian process experts. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1897–1904. Curran Associates, Inc., 2009.

Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Trans. Neural Netw. Learning Syst.*, 23(8): 1177–1193, 2012.

Shi-cang Zhang, Jian-xun Li, Liang-bin Wu, and Chang-hai Shi. Amultiple-maneuvering targets tracking algorithm based on a generalized pseudo-bayesian estimator of first order. *Journal of Zhejiang University SCIENCE C*, 14(6):417–424, 2013.