# Continuous States Conditional Random Fields Training Using Adaptive Integration


Submitted by

João José Amaro Amador Leitão

to the
University of Exeter
as a thesis for the degree of

*Doctor of Philosophy in Computer Science*

December 2010

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

.....

*This is not the end. It is not even the beginning of the end.*

*But it is, perhaps, the end of the beginning.*

Sir Winston Churchill

**To my family and friends.**

# Abstract

The extension of Conditional Random Fields (CRF) from discrete states to continuous states will help remove the limitation of the number of states and allow new applications for CRF.

In this work, our attempts to obtain a correct procedure to train continuous state conditional random fields through maximum likelihood are presented. By deducing the equations governing the extension of the CRF to continuous states it was possible to merge with the Particle Filter (PF) concept to obtain a formulation governing the training of continuous states CRFs by using particle filters. The results obtained indicated that this process is unsuitable because of the low convergence of the PF integration rate in the needed integrations replacing the summation in CRFs. So a change in concept to an adaptive integration scheme was made.

Based on an extension of the Binary Space Partition (BSP) algorithm an adaptive integration process was devised with the aim of producing a more precise integration while retaining a less costly function evaluation than PF. This allowed us to train continuous states conditional random fields with some success.

To verify the possibility of increasing the dimension of the states as a vector of continuous states a scalable version was also used to briefly assess its fitness in two-dimensions with quadtrees. This is an asymmetric two-dimensional space partition scheme.

In order to increase the knowledge of the problem it would be interesting to have further information of the relevant features. A feature selection embedded method was used based on the lasso regulariser with the intention of pinpointing the most relevant feature functions indicating the relevant features.

# Acknowledgements

I would like to thank my supervisor Professor Richard Everson, Motorola and EPSRC for the opportunity given to me to do a PhD, Professor Dragan Savic, Doctor Arnaud Marmier and Elizabeth Roberts for their help, Doctor Martin Jenkins and Doctor Neil Sewell for their good advice, Theodoros Economou and Richard Fredlund for our stimulating discussions and Edward Hirons for being a good friend.

I am also grateful to my family for their unconditional support even at moments when they were the ones in need of support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Limitations of existing models

The impact of recent developments in stochastic models, like Conditional Random Fields (CRF) [Lafferty 01] or Particle Filters [Doucet 08], has made a big contribution to the efficiency in the way which we deal with some current problems, something not achieved by more classical approaches. Classical approaches have been able to deliver solutions to a number of problems but have never been able to cope with certain areas such as automatic speech recognition, automatic translation, computer vision just to cite a few. The fast developments in this new field and its early successes, in conjunction with the nature and difficulties of the problem, led to an explosion of possible applications. The simplifications assumed in certain models, even if acceptable for a first order approximation, become apparently restrictive in these models. This issue is not new; deterministic theories are often unsuitable to handle real problems beyond a certain degree of precision. So in general, initial theories are replaced by more precise but, understandably, more complex ones. An example of this would be Einstein's theory of relativity replacing Newton's theory since Newton was unable to correctly predict the precession of Mercury (and more issues).

The issues of limitation are often very similar even between vastly different models in computer science; these limitations are mainly related to the number of calculations required to perform a given task, as opposed to the lack of mathematical tools available. Both limitations are somewhat connected, even if there is no mathematical tool able to

solve Navier-Stokes equations, we manage to approximate solutions by numerical methods but the complexity of the problems possible is bounded by the restriction due to the number of calculations limitation. A lot of research efforts are currently put into developing theoretical backgrounds which will allow existing models to perform better, or to work with more demanding applications. In this thesis we will explain how we tried to extend an existing model to accommodate new features while retaining an acceptable computational time. This model has a number of interesting applications like computer vision [Quattoni 07], robotics [Limketkai 07] or automatic speech recognition [Zweig 09]. Never the less our aim is not to apply the model to any particular application but to extend the range of the model so it will address a more broad class of applications.

The model we are going to address, Conditional Random Fields (CRF) [Lafferty 01], had its focus in treating time series data. We assume that time series are a sub-class of ordered sequences. Where ordered sequences are only indexed by a set with an axiom of order, time series are related to the index. This distinction is important since a model may address one of the classes but not the other. As opposed to a theory, a model does not have to have its intrinsic properties mimicking a process or system; this has the advantage of being independent from the process and in consequence might be successful in other processes. The major disadvantage is the model only being able to give us little, if any, information about the process. The conceptual representation of real world processes is defined as systems. So a model does not recreate systems but it is important to know in which types of systems a model is relevant. A model addressing ordered sequences, but not time series, might not be suitable to address time series with good results and vice-versa. A number of processes may be represented by a class of systems in which the internal dynamics produce results we gather as data. It is though this data that it is used to model the system.

The past history of a system is one of its important features, as is also how much of the past information is relevant. The information about the past might be not be evident; for example in systems modeled by a linear differential equation the information is carried by the derivatives, but these systems can be transformed from an $n$ order differential equation in to a system of $n$ equations of order one, and in consequence changing the way this information is carried.

The type of data gathered from the process is also relevant since it might represent an analog system in the case of a continuous variable or a digital system if the variable takes only discrete values. In theory, analog systems can carry an infinite amount of information, in practice however this is not possible, nevertheless to understand some processes is some times more advantageous to assume a continuous system.

One very successful model, the Hidden Markov Models (HMM) [Rabiner 89], which we will present later in more detail, only addresses a certain kind of system. It is limited to discrete systems and is unable to properly treat transient behaviour, since it deals with ordered sequences which are not time series. Another successful model, also briefly presented later,is the Kalman filters [Kalman 60], which even though were designed for analog systems, are limited to ordered sequences which are also not time series. Neither one of these models gives any information about the systems representing the process from which the data was collected.

There is a model on which we will focus our attention, that has a good prospect of being able to reach good results in several of these classes of system. Conditional Random Fields have some interesting properties and we think they will achieve better results than other models in some problems which represent the intersection of this classes.

The usual approach for the definition of these models is by using a probabilistic framework. This view is based on historical reasons. In the case of Conditional Random Fields however, a different view based only on algebraic arguments might give us a changed perspective. For example, under a probabilistic perspective the main difference between HMM and CRF is in the way they represent the choice of the the most probable next element of the sequence of variables of the system. The training process for HMM generates a function representing the total probability for the value of the element at of the unknown sequence the current time and the current value of the known sequence. The literature emphasises the fact that CRF does not generate this total probability but generates a conditional probability of the unknown sequence given the known sequence. Under a strict functional perspective, the resulting sequence obtained either by a conditional probability or by the total probability is only function of the known sequence and the model.

Some of the training methods we are going to use have been previously used with some success in other models, and CRF itself has used with success under certain conditions; so

we are interested in finding the behaviour of the model under these methods. In fact the Hidden Markov Models have preceded Conditional Random Fields so techniques to train HMMs to particular problems are available. Due to the similarity of the two models we will use the same techniques to train CRFs to particular problems.

## 1.2 Purposed extensions

Our perception of how to use the CRF model does not coincide with the previous approaches cited in the literature. In our opinion it would be possible to define a training method which allows a free choice of values for the internal properties in the training process of the model based on its relevance. A pertinent question is also if it would be feasible to perform this training by using an optimization algorithm which allows faster convergence by using the derivative of the optimized function. Lastly it will be interesting to know if it is possible to acquire more information about the system by using a mechanism which indicates which of the properties in the system are most relevant.

One usual approach for training models is by defining a function, the *likelihood*, using the parameters defining the model as variables to obtain the values of the parameters. This means using the variables to match the counting of the relevant features on the data obtained previously or training data and the expected number of features for the model for a set of parameters.

The difficulty in manipulating continuous functions in a computer, since computers do not work with concepts but with numbers, lies with how to represent these functions. If the models have interesting properties it is possible to obtain general results where the computations are only needed for the parameters governing the model. When this is not possible, the need for this representation might be overcome by some representative points. Particle Filters (PF) [Doucet 08] is one method which allows this representation. Due to the particularity of the model and since PF has also the advantage of the intrinsic property of allowing easy approximation of integrals, it is a good candidate to represent the relevant functions in the CRF model.

Another interesting way of generating points which are representative is by using Binary Space Partition (BSP) [Fuchs 80] in conjunction with a definition of the represen-

tativeness. So in conjunction with adaptive integration this allows the representation of continuous functions and the calculation of integrals in a relatively correct way. One advantage of BSP is that it can be easily extended to higher dimensions.

In real applications when the sequences are corrupted by noise there is a tendency for the parameters to fit not only the general behaviour of the system but also the noise. To prevent this behaviour it is common to add a penalty factor penalising this trend on the parameters. Some functions used as this penalty factor have a side consequence of also penalising the less relevant parameters by decreasing them. This has been tried successfully in CRF [Vail 08] so it is likely to produce the same result in a extension of the CRF to unknown sequences with values ranging outside a countable number of possibilities.

It is our aim to use the presented ideas to perform CRF training for the continuous state case in higher dimensions, and to assess which are the relevant feature by forcing the parameters affecting the irrelevant ones to zero.

## 1.3 Novel Contributions

In this work we have:

- Obtained a representation of the log-likelihood for Continuous State Conditional Random Fields and its gradient by algebraic methods. (Chapter 3)
- Found arguments to support the reason of the failure of the training of continuous state CRFs with particle filters. (Chapter 3)
- Introduced adaptive integration with Binary Space Partition and a scalable form to Conditional Random Fields training. (Chapters 4 and 5), and extend it to Ternary Space Partition
- Tested with success the Lasso regulariser for continuous states CRFs (Chapter 6)

## 1.4 Work Structure

The thesis is organised as follows:

**Chapter 2** -We will present some previous background work on using a CRF model or on the similar models of HMM and Kalman Filters to put it in context, emphasising the similarities and differences between them. We also present a method for training we will be using in a later stage .

**Chapter 3** -In this chapter we present our attempts for training the model, using the particle filters method presented in chapter 2, we present the description of our efforts, the results obtained and the conclusions we found.

**Chapter 4** - Based on the previous chapters conclusion, we change the method of training by using a different type of integration process based on the adapting this integration in function of the information we received from the integrand. We explain some of it interesting properties and limitations and we apply it to the training the model.

**Chapter 5** - An extension to higher dimensions for the continuous state CRF is attempted in this chapter, using a two dimension example. This extension present new challenges which will be presented, like the integration process. We will explain how the integration process presented on the previous chapter was extended.

**Chapter 6** - This chapter is dedicated to testing the possibility of having more insightful information on the choices of our model, by obtaining information on which features of the model are more relevant.

**Chapter 7** - Finally in this chapter we summarise the work done in the previous chapters draw the general conclusion and give some indication on the possible lines of research and indicate some points we were not able to fully clarify to our satisfaction.

# Chapter 2

# Background

## 2.1 Introduction

Since the correspondence between Pascal and Fermat, accepted as the start of probability and statistics studies as in [Hald 90], we have thought that our attempts to deal with uncertainty can be rationalised. For a relatively long time now, the scientific community has tried to overcome uncertainty using the formal tools given to us, by them. It is only recently that we have managed to see this opportunity to use stochastic methods as a way of tackling deterministic problems that can otherwise be considered infeasible with our current state of technology. Of course the price to pay for this is that, in most cases, we are only able to achieve a sub-optimal solution. That is, solutions that are close to the "true" value but which are not the "correct" solution in formal terms but which are good enough to be acceptable in our uncertain world.

Engineers have dealt with uncertainty for a long time, they knew that they did not know the exact solution but bridges and trains still had to be built and improved, thus they managed to accommodate uncertainty in their projects; the scientific correctness of a solution was only marginal in relation to the search for "higher, faster or cheaper" as [Shigley 86] states. Under the pressures of evolution this idea spread to other fields, such as computer science, where the deterministic approach has since been overshadowed, in a number of applications, by a looser but efficient way of obtaining answers and solutions. For example, for higher dimensions integration we have seen an increased use of Monte Carlo integration as opposed to more deterministic methods.

With the development of more powerful and faster computers, not only have new ways of dealing with older, unsolved, problems been devised and new methods developed to improve existing solutions, but also new problems have been created by the same application.

In this chapter, probabilistic methods for inferring information about sequence data are presented, where the sequence data variable of interest is not directly observed, a so-called "hidden" variable, but rather a probabilistically related measurement or observed variable is known. We can formulate our model by thinking of the hidden variable as a *state* which determines the observation.

We denote this state vector by $\mathbf{x} = \{x_0, x_1, \ldots, x_T\}$ for a sequence of $T+1$ states, i.e. at each time, $t$, components are given by $x_t$ and may be either discrete or continuous. We can similarly formulate the observation vector $\mathbf{y} = \{y_0, y_1, \ldots, y_T\}$. These variables $x_t$ and $y_t$ can be multidimensional with $N$ and $M$ denoting the dimensionality of the state and observation spaces respectively, if required.

The model is completed with an assumption of how the state at time $t$ depends on previous states and a model of how observations are related to states. Given state transition and observation models, the goal of inference for sequence models is thus to infer the values for "$\mathbf{x}$" having made observations "$\mathbf{y}$". If the parameters of the models, which are in general denoted by $\lambda$ are unknown, then the principal goal of learning in sequence models is to find good values for $\lambda$, usually by maximizing a likelihood.

All the models we will examine employ the first-order Markov assumption that the available knowledge about a state from prior states is complete when the state of its predecessor is known; that is

$$p(x_t|x_{t-1}, \ldots, x_0) \equiv p(x_t|x_{t-1}) \tag{2.1.1}$$

Our main interest will be focused on models with continuous states and observations but we review discrete state Hidden Markov Models (HMM) in section 2.2.1 because many of the ideas needed for linear chain Conditional Random Fields (CRF) are conveniently formulated for the more familiar HMM.

In the next section we review discrete state and observation HMM's and CRF's after

which we move onto continuous models. In section 2.4 we outline the particle filter methods for stochastic integration.

## 2.2 Sequence Classifiers

Assigning objects into classes or "sets", based on their properties, is done by classifiers. In the algorithmic case these classifiers take a numerical input as a number, or a set of numbers, to produce a classification represented by a discrete label. In the case of the continuous output value, this is then called *regression*. A classification may be produced by a decision inherent to the model, in which case it is the model which adjusts itself to comply to its own and externally defined constraints, and in this case classification is unsupervised, usually known as *clustering*. Alternatively it can be taught to mimic a number of classifications decisions, by supervised learning, where a set of inputs and classes are supplied for training purposes.

We will assume as classifiers all the class of methods of classification, sequence classifiers the methods which its main applications is to classify sequences like the HMM, and non-sequence classifiers as the classifiers which don't belong to that class, like the $k$-mean clustering [Duda 01].

The main difference between the use of a sequence classifier and a non-sequence classifier used sequentially on elements of a sequence, is that a sequence classifier improves its probability of success by taking into account the conditional probability of several sequential elements in the sequence. In theory if the elements of the sequence are in some way statistically correlated, the probability of correct classification using this information is at least equal to the probability of using each element of the sequence alone. In a sense, a sequence classifier splits the problem of classification of a sequence into a classification of each element on its own, with additional information obtained by knowledge of the statistical relationship between the data [Dietterich 02].

### 2.2.1 Hidden Markov Models

The Hidden Markov Model (HMM), is probably the most well known sequence classifier and is also known as a non-deterministic finite state machine, and has several well doc-

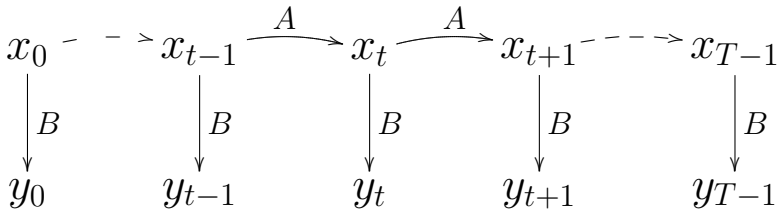


Figure 2.1: Hidden Markov Model transition relations.

umented applications in different fields such as: speech recognition [Rabiner 89], stock market forecasting [Hassan 05], etc ...

We can present HMM as

$$p(x_t|x_{t-1}, \ldots, x_0, y_t) \equiv p(x_t|x_{t-1}, \ldots, x_{t-i}, y_t) \tag{2.2.1}$$

where the first $n$ terms (where $n$ represents the order of the HMM) are the conditional probability factors from the sequence correlation and the last factor is the probability of a correct classification based on the current observation. The order of the HMM is a consequence of the Markov assumption which states that a given state is only statistically related to the $n$ previous states and the current observation. For a HMM of order one a state at time $t$ is only a function of the observation at time $t$ and the state at time $t-1$, the previous state. HMMs only deal with discrete states, and the cardinal of the set of all the possible states is finite.

Let us assume an HMM of order one with parameters $\lambda = (A, B, \pi)$, where $A$ is a matrix storing the probability of transition from state $x_i$ to state $x_j$ ($A_{ij} = p(x_i|x_j)$), $B$ is the matrix for the probabilities of the emission of the observation $y_j$ given the state $x_i$ ($B_{ij} = p(y_i|x_j)$), and $\pi$ is the initial distribution of the state for $t = 0$ or initial conditions. As summarised in Figure 2.1, each state is related only to the previous state through the matrix $A$, as a consequence of the Markov assumption, and with the current observation through the matrix $B$.

Three problems have been defined by [Rabiner 89], which summarise the intrinsic strengths and difficulties of HMM. We describe these problems and their solutions.

**Problem 1 : Likelihood** *Considering a HMM of order* 1 *with parameters* $\lambda = (A, B, \pi)$ *and a sequence of observations* $\boldsymbol{y}$*. What is the likelihood of the sequence* $\boldsymbol{y}$ *given the parameters* $\lambda$ *?*

Let us suppose that we have a state sequence $\mathbf{x}$ of length $T$ the same length as the sequence of observations, then by the definition of $A$ and $B$, the probability of the observations given the state sequence and the parameters is:

$$p(\mathbf{y}|\mathbf{x}, \lambda) = b_{x_0}(y_0)b_{x_0}(y_0)\dots b_{T-1}(y_{T-1}) \tag{2.2.2}$$

and the probability of the state sequence given $\lambda$ is

$$p(\mathbf{x}|\lambda) = \pi_{x_0}a_{x_0x_1}a_{x_1x_2}\dots a_{x_{T-2}x_{T-1}} \tag{2.2.3}$$

By the conditional probability formula we know that

$$p(\mathbf{y}, \mathbf{x}|\lambda) = p(\mathbf{y}|\mathbf{x}, \lambda) \cdot p(\mathbf{x}|\lambda) \tag{2.2.4}$$

then we will have, defining $\mathbf{X}$ as the space of all sequences of states, by doing the summation on all the possible sequence of states:

$$\begin{aligned} p(\mathbf{y}|\lambda) &= \sum_{\mathbf{X}} p(\mathbf{y}, \mathbf{x}|\lambda) \\ &= \sum_{\mathbf{X}} p(\mathbf{y}|\mathbf{x}, \lambda) \cdot p(\mathbf{x}|\lambda) \\ &= \sum_{\mathbf{X}} \pi_{x_0}b_{x_0}(y_0)a_{x_0x_1}b_{x_1}(y_1)\dots a_{x_{T-2}x_{T-1}}b_{x_{T-1}}(y_{T-1}) \end{aligned} \tag{2.2.5}$$

The issue arising with this computation is that it will became intractable as soon as the number of states or the length of the sequence increases because it will require $2T \cdot N^T$ multiplications, where $N$ is the number of states. So a new method must be devised.

Let us define the joint probability of each state $s_i$ given a sub-sequence of observations from 0 to $t$ given the parameters

$$\alpha_t(s_i) = p(\mathbf{y}_{0,\dots,t}, x_t = s_i|\lambda) \tag{2.2.6}$$

so now we can derive a recurrence formula for $\alpha$. The recurrence formula for $\alpha$ can be derived as follows

1. The probability for the first sub-sequence will be: $\alpha_0(s_i) = \pi_o b_0(y_0)$

2. Then for each time step added, we can write recursively

$$\alpha_t(s_i) = \left[\sum_{j=0}^{N-1} \alpha_{t-1}(s_j) a_{ji}\right] b_i(y_t) \tag{2.2.7}$$

So the total probability $p(\mathbf{y}|\lambda)$ is given by the summation on all the states for the full sequence:

$$p(\mathbf{y}|\lambda) = \sum_{i=0}^{N-1} \alpha_{t-1}(s_i) \tag{2.2.8}$$

The advantage of this change is that it reduces the need for $2T \cdot N^T$ multiplications to only $N^2 \cdot T$, and so transforms an intractable calculation for high numbers of states and sequence length, to a tractable algorithm not limited to a small number of states or time steps.

**Problem 2 : Sequence of most probable states** *If we consider a HMM of order* 1 *with parameters* $\lambda = (A, B, \pi)$ *and a sequence of observations* $\boldsymbol{y}$*. What is the sequence of the most probable states* $\boldsymbol{x}$ *given the sequence* $\boldsymbol{y}$ *and the parameters* $\lambda$ *?*

Let us note first that the sequence of most probable states is not the same as the most probable sequence, because a state at time $t$ may be the most probable state but may not be in the most probable sequence. We will see later in section 2.2.4 how to find the most probable sequence.

We have seen how to calculate the probability of a given state for a sub-sequence of observations $\mathbf{y}_{0,\ldots,t}$, so we can now, through a similar process, find the probability for each state at time $t$ for the observations sub-sequence $\mathbf{y}_{t,\ldots,T-1}$. We define $\beta_t(s_i) = p(\mathbf{y}_{t,\ldots,T-1}, x_t = s_i|\lambda)$, then

1. The probability for the first sub-sequence starting in the opposite direction will be: $\beta_{T-1}(s_i) = 1/N$. We have to note the difference between this and the $\alpha$ expression (2.2.6). We have defined a starting probability $\pi_i$ but this is only valid in the

forward direction, so we have to assume the ignorance of the initial distribution for the backward direction and assume an equal probability distribution.

2. Again for each time step we can write recursively:

$$\beta_t(s_i) = \sum_{j=0}^{N-1} a_{ji} b_j(y_{t+1}) \beta_{t+1}(s_j) \tag{2.2.9}$$

So we can now calculate the probability of any state at a given time by

$$\begin{aligned} \gamma_t(s_i) = p(x_t = s_i|\lambda) &= \frac{p(x_t = s_i, y_0, \ldots, y_t|\lambda) p(x_t = s_i, y_t, \ldots, y_{T-1}|\lambda)}{p(\mathbf{y}|\lambda)} \\ &= \frac{\alpha_t(s_i)\beta_t(s_i)}{p(\mathbf{y}|\lambda)} \end{aligned} \tag{2.2.10}$$

As a consequence the solution to our second problem is given by

$$\hat{x}_t = \underset{s_i}{\operatorname{argmax}}\, p(x_t = s_i|\lambda) = \underset{s_i}{\operatorname{argmax}}\, \gamma(s_i) \qquad \text{for } t \text{ in } [1, T-1] \tag{2.2.11}$$

which will generate the sequence of most probable states. The calculation of the $\alpha$ and $\beta$ pass is known as the forward-backward algorithm [Rabiner 89].

**Problem 3 : Likelihood maximisation** *For a HMM with a sequence of observations* $\boldsymbol{y}$ *and the dimensions* $N \times N$ *and* $N \times M$ *of the matrices* $A$ *and* $B$*, what are the values of* $A$*,* $B$ *and* $\pi$ *which maximize the probability of the sequence of observations* $\boldsymbol{y}$ *?*

The solution for our third problem will be given by the *Baum-Welch algorithm*, which is the application of the *Expectation-Maximisation* (EM) algorithm [Baum 70], [Dempster 77], [Bishop 07] to HMM. In this case, as opposed to our previous problem, we need to calculate a transition probability, so we need to define a new $\gamma$ function taking into account two states by

$$\xi_t(s_i, v_j) = p(x_t = s_i, x_{t+1} = v_j|\mathbf{y}, \lambda) = \frac{\alpha_t(s_i) a_{ij} b_j(y_{t+1}) \beta_{t+1}(v_j)}{p(\mathbf{y}|\lambda)} \tag{2.2.12}$$

By definition $\gamma$ and $\xi$ are related by the expression

$$\gamma_t(s_i) = \sum_{j=0}^{N-1} \xi_t(s_i, s_j) \tag{2.2.13}$$

and we clearly see that the initial distribution $\pi$ will be given by $\pi_i = \gamma_0(s_i)$. Each element of the $A$ matrix, incorporating the transition probability from state $s_i$ to state $s_j$ is obtained by the following expression

$$a_{ij} = \frac{\sum_{t=0}^{T-2} \xi_t(s_i, v_j)}{\sum_{t=0}^{T-2} \gamma_t(s_i)} \tag{2.2.14}$$

where the numerator represents the frequency of that particular transition (between the states $s_i$ and $s_j$) and the denominator represents the frequency of the state $s_i$ so by consequence we obtain the relative frequency for that particular transition.

To obtain the values of $B$ we need to compute the relative frequency of each observation by

$$b_j(s_i) = \frac{\sum_{\substack{t=0 \\ y_t = o_j}}^{T-2} \gamma_t(s_i)}{\sum_{t=0}^{T-2} \gamma_t(s_i)} \tag{2.2.15}$$

in which the numerator is the frequency of a given state with a given observation $o_i$, and the denominator is the frequency of that same state.

Now that we can find estimates of the values for the matrices $A$ and $B$, we need to iterate the process as follows;

1. Initialize the parameters $\lambda = (A, B, \pi)$ with some sensible guess or with an equi-probable distribution (as previously done for the beta pass).

2. Calculate $\alpha_t(s_i)$, $\beta_t(s_i)$, $\xi_t(s_i, s_j)$, and $\gamma_t(s_i)$ using the forward-backward algorithm (E-step).

3. Re-evaluate the values of the matrices $A, B$ and the vector $\pi$ to obtain a new set of parameters $\lambda = (A, B, \pi)$ (M-step).

4. Compute the value for $p(\mathbf{y}|\lambda)$ and compare it with the value from the previous iteration, if it has converged to within a small region of the previous $\lambda$ then we can stop since we have found a close approximation of our parameters, otherwise we make another iteration by jumping back to step 2.

With this iteration process, at each time step we will get closer to the true values of the parameters for our training data, the proof of this convergence can be found in [Borman 04].

One of the problems with the forward-backward algorithm is, even with a sensible number of states and time steps, the values in $\alpha$ and $\beta$ can grow very quickly. To prevent numerical instabilities, like overflows, we can "unload" these values by normalisation. This normalisation is computed by the summation of $\alpha_t$ on every state as

$$\hat{\alpha}_t = \sum_{i=1}^{N} \alpha_t(s_i) \tag{2.2.16}$$

which allows us to define a sequence of scaling factors

$$\breve{\alpha}_t(s_i) = \frac{\alpha_t(s_i)}{\hat{\alpha}_t} \tag{2.2.17}$$

These scaling factors can be stored in logarithmic form, further improving the numerical stability.:

$$\breve{\alpha}_t = \breve{\alpha}_{t-1} + \ln(\hat{\alpha}_t) \qquad \text{with} \qquad \breve{\alpha}_0 = 0 \tag{2.2.18}$$

Thus, clearly alpha will be

$$\alpha_t(s_i) = \breve{\alpha}(s_i)\ \exp(\breve{\alpha}_t) \tag{2.2.19}$$

and the calculation of $\xi$ and $\gamma$ can be conducted with the help of the properties of the logarithmic and exponential functions.

The issue arising with HMM of orders higher than one is the number of variables required to describe the model. We have previously stated that the order of the Markov model is defined solely by the number of previous consecutive elements in the sequence considered by the statistical analysis. The transitions probabilities for a first-order finite state model can be represented by a matrix with dimension $N \times N$, where $N$ is the number of admissible states. For a higher order, it will require a tensor of order $W$ with

dimension $N^{W+1}$, where $W$ is the order of the Markov model. Another issue related to high-order HMMs is the number of training data required, with a second order HMMs with 10 states we need to calculate $10^3$ transition coefficients. It is clear to see that the size and number of the samples we need, to correctly train an approximate value of these transition probabilities, will increase unacceptably. Even with a limited number of states, the computable complexity of the models becomes very quickly intractable. For this reason the HMMs used in applications are mainly of order one.

HMMs are simple, quick and appropriate as a first approximation model but they also have some very strong limitations as a more precise approximation tool, especially first order models. To overcome this limitation while maintaining the HMM's strengths, some authors use alternative formulations, such as the layered HMM [Oliver 04] or the semi-Markov HMM [Guedon 03]. Nevertheless, none of these models can produce good enough approximations on long term correlations without an excessive computational complexity.

### 2.2.2 Conditional Random Fields

Conditional Random Fields (CRF) are a stochastic model based on applying probabilities to graph models and are defined in acyclic graphs. Since extensive studies on graph relations are available, based on the search for their own intrinsic proprieties, changing the relation between nodes to a probability allows us to generate new stochastic models, where some well known results from the graph study can then be applied [Murphy 01]. This was the path used by [Lafferty 01] to introduce CRF models. Conditional Random Fields have been introduced to overcome the *Label Bias Problem* where the labelling, in certain cases, can be inconsistent in Maximum Entropy Markov Models (MEMM) a good example is presented in [Lafferty 01]. As we said in previously a number of works have been published in recent years as Conditional Random Fields have been successfully applied in several areas: computer vision [Quattoni 07], [Vallespi-Gonzalez 08], [Tappen 07], [Taycher 06]; robotics: [Limketkai 07], [Vail 08]; automatic speech recognition: [Zweig 09], [Gregory 04]; information retrieval and parsing: [Sarawagi 04], [Sha 03], [Shen 07] just to cite a few authors and fields of application.

To the detriment of the historic approach, we will present CRF here under different perspective based on the maximum entropy principle, like a MEMM model, as in

[Klinger 07]. In our opinion this approach exposes more clearly that CRF are a more general model than HMM, their main difference is based on the possibility of the CRF being able to take account of more information from the observations sequence for a given state classification, since it is not limited to the one observation whose index matches the state models.

We obtain the Conditional Random Fields model by defining a limited number of constraints over the maximum entropy principle. The maximum entropy principle assures that a system will take the most stable or entropic state given the constraints [Callen 85]. This physical principle has been re-framed under a probabilistic framework by Boltzmann [Callen 85] and was transposed to information theory by Shannon [Shannon 48]. Three constraints have to be imposed. The first two are to ensure the compliance of the model to the probability axioms, since those limitations are not inherent from the set of the calculus axioms. The third, which is a stochastic argument, is that the expectation of some feature functions in all possible sequence of states given a sequence of observations must be as close as possible to the expectation of the feature functions in the data. Implicitly this last constraint assumes the existence of some relevant features inherent to the process and present in the data, and that it is possible to assess these features using a set of feature functions.

As opposed to a functional theory perspective, under the probability theory a relevant difference between CRF and HMM is now both models express the relations between states and observations. Conditional random fields is formally a conditional model as it defines the conditional probability $p(\mathbf{x}|\mathbf{y}, \lambda)$. On the other hand HMM model the joint probability $p(\mathbf{x}, \mathbf{y}|\lambda)$ of state and observation given the parameters.

Let us assume a finite family of feature functions $f_i(s, v, \mathbf{y}, t)$, or $f_i(\mathbf{x}, \mathbf{y})$ in a shorter notation, with $I$ functions, where $s, v$ are two consecutive states based on $t$ and $\mathbf{x}$ of the observation sequence. The arguments of the feature functions are; the previous state, the current state, the observation sequence, the current value of the time index. The feature functions represent numerically the relevance of features present in the sequence. Considering the training sample representative of the intrinsic statistical properties of the model we can assume that the expectation of the feature functions on the empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$, were the empirical distribution is the proportion of a given state in

the training data $\mathcal{Z}$,

$$\tilde{E}(f_i) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{Z}} \tilde{p}(\mathbf{x},\mathbf{y}) f_i(\mathbf{x},\mathbf{y}) \tag{2.2.20}$$

must be equal to the expectation of the feature functions on the model distribution,

$$E(f_i) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{Z}} p(\mathbf{x},\mathbf{y}) f_i(\mathbf{x},\mathbf{y}) \tag{2.2.21}$$

so

$$E(f_i) = \tilde{E}(f_i) \tag{2.2.22}$$

Also a normalisation must be made to ensure that

$$p(\mathbf{x} \mid \mathbf{y}) \geq 0 \quad \text{for all } \mathbf{x},\mathbf{y} \qquad \text{and} \qquad \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{x} \mid \mathbf{y}) = 1 \tag{2.2.23}$$

where $\mathcal{Y}$ is the set of all possible sequences, to conform with the probability theory axioms. Using Lagrange multipliers the variational functional becomes,

$$\Lambda(p,\lambda) = H(\mathbf{x} \mid \mathbf{y}) + \sum_{i=1}^{I} \lambda_i (E(f_i) - \tilde{E}(f_i)) + \lambda_{I+1} \left( \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{x} \mid \mathbf{y}) - 1 \right) \tag{2.2.24}$$

where the conditional entropy is given by

$$H(\mathbf{x} \mid \mathbf{y}) = - \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{Z}} p(\mathbf{x},\mathbf{y}) \log p(\mathbf{x} \mid \mathbf{y}) \tag{2.2.25}$$

This ensures that we find the most general distribution, of all the possible distributions given $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, the set of all possible inputs and outputs from the state and observations spaces, that are consistent with the information on the training data.

After finding the local maximum point through locating the zero of $d\Lambda/dp(\mathbf{y} \mid \mathbf{x})$, we obtain the following equation (see [Klinger 07] for further details),

$$p_\lambda(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp\left( \sum_{i=1}^{I} \lambda_i f_i(\mathbf{x},\mathbf{y}) \right) \tag{2.2.26}$$

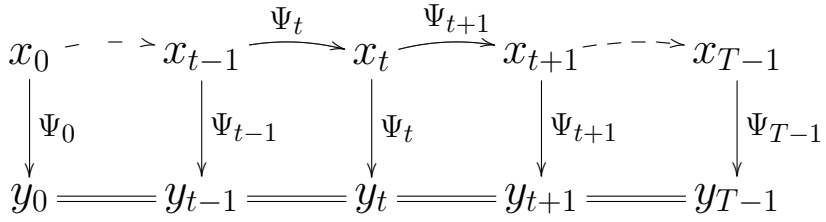


Figure 2.2: Conditional Random Fields transitions relations.

where the normalisation factor $Z_\lambda(\mathbf{x})$ is given by

$$Z_\lambda(\mathbf{x}) = \sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(\mathbf{x}, \mathbf{y})\right) \tag{2.2.27}$$

Also note that an implicit constraint comes from the $f_i$ function. The properties from these feature functions will define the type of model of data they will be able to analyse. Using feature functions which do not assess the features present in a given problem will produce unsuitable results. So for a given problem it is important to know which features are present and which are relevant. This might be an important issue if no prior information is available for the problem.

Contrary to other variational problems no other information is available to restrict the set of feature functions based on the required properties due to known information from relations to the data, like in the Galerkin or weak formulation, for the finite elements method for example [Zienkiewicz 05]. In the finite elements method the differential equation will impose more constraints on the functions to allow them to transform this intractable differential equation in a solvable one.

The representation of the relations in linear chain CRF models can be seen in Figure 2.2. As opposed to the HMM, the transition probability from state $s_i$ to $s_j$ is no longer only a function of both states and the current observation $p(x_t = s|x_{t-1}, y_t, \lambda)$, but it is also a function of all the other observations, represented by the double line connecting the observations $\mathbf{y}$. A second important point is that we do not have a probability as in the HMM case, but a potential $\Psi_t(s, x_{t-1}, \mathbf{y}, \lambda) = \exp(\sum f_i(s, x_{t-1}, \mathbf{y}, \lambda))$, proportional to that probability $p(x_t = s|x_{t-1}, \mathbf{y}, \lambda) \propto \Psi_t(s, x_{t-1}, \mathbf{y}, \lambda)$.

The strength of a Conditional Random Field model is the flexibility of its feature functions, virtually the only constraints on the feature functions are the ones imposed

by mathematical stability and the representativeness of the features in the model and in the training data. But the difficulty lies in that fact that it may be challenging, for a given problem, to define a correct set of feature functions. As an example, a set of feature functions based on

$$f_i(s', s, \mathbf{y}, t) = \begin{cases} 1 & \text{if } s = S_j \text{ and } s' = S_k \text{ and } y_t = O_l \\ 0 & \text{otherwise} \end{cases} \tag{2.2.28}$$

will generate a $1^{st}$ order HMM , but for more complex problems this might not be so clear. In this case the number of degrees of freedom for the weights $\lambda_i$ are the same as the number of degrees of freedom of the parameters in the HMM. Also the elements of the matrices for a HMM are bounded to the unitary interval which might not be the case for the weights in CRF. This is the reason why, in the case of CRF, there is a rescaling of the local potential by the total potential, but no certainty is available for different cases. We have to assume this ignorance about how to define relevant feature functions is a limitation of the CRF models, just as we assume the lack of knowledge about the number of states for a HMM model.

### 2.2.3 Training methods

To determine the correct parameters for a model, several general methods have been devised [Bishop 07]. We have explained in section 2.2.1 how to obtain those parameters in the particular case of HMM, through the Expectation Maximisation method. We will explain here how to find these parameters by maximising the log-likelihood for CRF. But other methods exist for maximum *a posterior* estimation and also have also been transposed for CRFs, for example Variational Bayes [Qi 05] and the Laplace method [Welling 06]. But we'll restrict ourselves here to the numerical solution for the Maximum Likelihood.

**Maximum Likelihood** The composition of a monotonic function with a given continuous function will not alter the location of the inflexion points for this function. As a consequence the multiplications involved in the likelihood can be transformed into simple additions by using the natural logarithm as the monotonic function. The conditional

likelihood, not the likelihood function $p(\mathbf{y}, \mathbf{x}|\lambda)$ as in the HMM case, is given by

$$\mathcal{L}(\lambda) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \log p(\mathbf{y} \mid \mathbf{x}, \lambda) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \log \left[ \frac{1}{Z_\lambda(\mathbf{x})} \exp\left( \sum_{i=1}^{I} \lambda_i f_i(\mathbf{x}, \mathbf{y}) \right) \right] \tag{2.2.29}$$

$$\mathcal{L}(\lambda) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \left( \sum_{t=1}^{T} \sum_{i=1}^{I} \lambda_i f_i(\mathbf{x}, \mathbf{y}) - \log Z_\lambda(\mathbf{x}) \right) \tag{2.2.30}$$

and again we need to find the local maximum with regard to $\lambda_i$. $I$ is the number of feature functions. Unfortunately equation (2.2.30) does not have a universal analytical solution easily obtained for all the possible feature functions so a numerical algorithm must be used. To improve the efficiency of the search we can use a Newton-type algorithm, which has been found to be faster than other types of numerical minimisation algorithm for CRFs [Wallach 04], and as a consequence we must find the gradient in relation to the $\lambda_k$. The relevant terms are:

$$\frac{\partial}{\partial \lambda_k} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{t=1}^{T} \sum_{i=1}^{I} \lambda_i f_i(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{t=1}^{T} f_k(\mathbf{x}, \mathbf{y}) \tag{2.2.31}$$

and

$$\begin{aligned} \frac{\partial}{\partial \lambda_k} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \log Z_\lambda(\mathbf{x}) &= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \frac{1}{Z_\lambda(\mathbf{x})} \frac{\partial Z_\lambda(\mathbf{x})}{\partial \lambda_k} \\ &= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \underbrace{\sum_{\mathbf{y}'\in\mathcal{Y}} p_\lambda(\mathbf{x} \mid \mathbf{y}')}_{\text{see equation 2.2.29}} \sum_{t=1}^{T} f_k(\mathbf{x}, \mathbf{y}') \end{aligned} \tag{2.2.32}$$

Our aim here is to calculate the conditional probability $p_\lambda(\mathbf{x} \mid \mathbf{y}')$ of a given state sequence, given $\mathbf{y}'$ and $\lambda_k$. Again, the number of possible state sequences are exponential and for solving this particular problem there is the possibility of using an algorithm similar to the one used as part of the HMM training, namely, the $\alpha$-$\beta$ recursion (or the forward-backward procedure). This is because by using the distributive law we can transform a summation of a set of multiplications into a recursion of summations.

We can see from equation (2.2.29) that the transition "probability" is given by the exponential of a summation of the feature functions $f_i$ weighted by the $\lambda_i$ coefficients. So

the transition potential from state $s_{t-1}$ to state $s_t$ is given by

$$\psi_t(s_t, s_{t-1}, \mathbf{y}, t) = \exp \sum_{i=0}^{I} \lambda_i f_i(s_t, s_{t-1}, \mathbf{y}, t) \tag{2.2.33}$$

Since the probability of the state sequence given the observation $\mathbf{y}$ and the model parameters $\lambda$ is

$$p(\mathbf{x} \mid \mathbf{y}, \lambda) = \prod_{t=1}^{T} p(x_t \mid y_t, \lambda) \tag{2.2.34}$$

and if we define $\alpha_t(j)$ and $\beta_t(j)$ as

$$\alpha_t(j) = \sum_{s_i \in \mathcal{S}} \alpha_{t-1}(i) \psi_t(s_j, s_i, \mathbf{y}, t) \tag{2.2.35}$$

$$\beta_t(j) = \sum_{s_i \in \mathcal{S}} \beta_{t+1}(i) \psi_t(s_i, s_j, \mathbf{y}, t) \tag{2.2.36}$$

with initial conditions

$$\alpha_0(j) = \beta_{T+1}(j) = 1 \tag{2.2.37}$$

with $S$ being the set of the possible states, then the derivative of the normalising factor (equation (2.2.32)) becomes

$$\begin{aligned} &\frac{\partial}{\partial \lambda_i} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \log Z_\lambda(\mathbf{x}) = \\ &\qquad \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \frac{1}{Z_\lambda(\mathbf{x})} \sum_{t=1}^{T} \sum_{k=1}^{n} \sum_{l=1}^{n} f_i(\mathbf{x}, \mathbf{y}) \ \alpha_t(k) \ \psi_t(s_l, s_k, \mathbf{x}, t) \ \beta_{t+1}(l) \end{aligned} \tag{2.2.38}$$

It's important to note that the $\alpha$ and $\beta$ and the summation on $k$ and $l$ can and should, in *stricto sensu*, be restricted to the possibilities of the distribution, since a given state can only admit a set of predecessors $\Gamma(s) \subseteq \mathcal{S}$.

### 2.2.4 Inference and the Viterbi algorithm

Based on the trained model, the derivation of the most probable sequence, given the observations, is obtained by finding the most probable path through all the possible sequences of states. An exhaustive search becomes quickly computationally intractable as soon as

we increase the number of states or the number of time steps. Nevertheless a strategy can be devised to overcome this problem using the Viterbi algorithm [Viterbi 67]. The Viterbi algorithm is based on the Bellman's principle of optimality [Lewis 95]. Bellman's principle states that each sub-path of the optimal path is an optimal path on its own. For that reason the Viterbi algorithm only needs to, for each state at a given time index, calculate the maximum total probability based on the probability of the previous states, and to store this probability and the state generating it to be used by the posterior state. By this strategy to build the optimal path as the time index increases, the Viterbi algorithm achieves the maximisation of the total probability for a sequence in only $TN^2$ operations. The algorithm is closely related to the alpha recursion, but instead of retaining all the values we are only interested in its maximum. So for a $1^{st}$ order HMM the process can be summarised as

1. Initialize $\delta_0(s_i) = \pi_i b_i(y_0)$ for all states, and $\zeta = \{\,\}$ with the empty set.

2. For every state at every time step

$$\delta_t(s_j) := \max_{0 \leq i \leq N-1} [\delta_{t-1}(s_j) a_{ji} b_i(x_t)] \tag{2.2.39}$$

   and

$$\zeta := \zeta + \{ \operatorname*{argmax}_{0 \leq i \leq N-1} [\delta_{t-1}(s_j) a_{ji} b_i(x_t)] \} \tag{2.2.40}$$

3. The termination is given by the most probable sequence sequence where

$$\zeta := \zeta + \{ \operatorname*{argmax}_{0 \leq i \leq N-1} [\delta_{T-1}(s_j) a_{ji} b_i(x_t)] \} \tag{2.2.41}$$

   with probability

$$p(\mathbf{x}|\mathbf{y}, \lambda) = \max_{0 \leq i \leq N-1} [\delta_{T-1}(s_j) a_{ji} b_i(x_t)] \tag{2.2.42}$$

The extension to CRF can be achieved with little effort, by changing the transition probability given by $a_{ji} b_i(x_t)$ by the potential $\Psi_t(s_j, s_i, \mathbf{y}, \lambda)$. Since $\Psi_t$ is proportional to the probability and this is invariant through the max and argmax operations.

## 2.3 Continuous States and Observations

Models for analysing sequences based on the Markov assumption of dependence of the states from only the $n$ previous states, are not restricted to discrete distributions both for states and observations, they are also valid for continuous ones. But as in the discrete case, as soon as the order of the model increases, the difficulty to handle these models also increases. In the continuous observations case the generalisation is relatively easy. We will be dealing with the full generalisation to continuous observations and states.

In the case of Markov models of order one, the limitation to a finite set of states both in HMM or CRF for example, have been removed either with a Kalman filter for HMM and with Continuous-CRF or CRF-Filters by [Taycher 06] and [Limketkai 07] for the CRF model, to a continuous set at least isomorphic to $\mathbb{R}^n$. We will further address Continuous States Conditional Random Fields in the next chapter.

### 2.3.1 Linear Quadratic Estimator

The concept behind a Kalman filter, the extension to Gaussian distributions of the HMM model, also known as Linear Quadratic Estimator (LQE), is defined as a stochastic linear filter. So by definition the Kalman filter defines, at each time step, a Gaussian distribution around a mean value.

Kalman filters can be defined as a linear system on a state space with Gaussian noise [Kalman 60], where the observations are also linear function of the state and have Gaussian noise. This allows the inference to be executed recursively and it is why Kalman filters can be seen as part of adaptive control. The algorithm of inference for Kalman filters can be deduced by minimizing the error between the real state and the predicted state. It unfolds in two steps, a predicting phase and an updating phase, the predicting phase is based only on the previous state and the update stage is based on the observation. The difficulty is to maintain a correct approximation of the distribution, which is handled by using a covariance matrix as a dual of the state. Identification of the complete linear system can be done by an Expectation Maximization algorithm.

It is a well known field, so an extended bibliography can be found on the subject [Ogata 87] or [Friedland 96] from a control perspective and [Bishop 07] for a more statis-

tical view. We will restrict ourselves here to a brief overview.

Let us see the one-dimensional example given by [Schutter 99]. Assuming that if only two independent observations $y_0, y_1$ of the same variable $x$ contaminated with Gaussian noise with zero mean and variance $\sigma_0^2, \sigma_1^2$ are available, then we can assume that a weighted mean of this observations will give us the best value of the estimate $\hat{x}$ of the variable $x$:

$$\hat{x} = w\, y_0 + (1 - w) y_1 \tag{2.3.1}$$

We assume that $x$ has a Gaussian distribution $\mathcal{N}(\hat{x}, \sigma^2)$ then its standard deviation will be given by $\hat{\sigma}^2 = w^2\, \sigma_0^2 + (1 - w)^2 \sigma_1^2$ as a consequence of the assumption of the distribution of each observations are Gaussian. By finding the value which maximises $\hat{\sigma}^2$ we can find the best weight, since it's.

$$\frac{d}{dw}[w^2\, \sigma_0^2 + (1 - w)^2 \sigma_1^2] = 0 \qquad \Rightarrow \qquad w_{opt} = \frac{\sigma_1^2}{\sigma_0^2 + \sigma_1^2} \tag{2.3.2}$$

Now by assuming that the two observations are sequential, we can now have a one-dimension linear recursive estimator. Let us make $\hat{x}_t = y_t$ and $\hat{\sigma}_t = \sigma_0$ and $K = \sigma_t^2/(\sigma_{t-1}^2 + \sigma_t^2)$, then

$$\begin{aligned} \hat{x}_t &= \hat{x}_t + K(y_t - \hat{x}_{t-1}) \\ \sigma_t^2 &= (1 - K)\sigma_{t-1}^2 \end{aligned} \tag{2.3.3}$$

So we have obtained a recursive expression for evaluation of the state $\hat{x}$ given two sequential observations $y_{t-1}, y_t$ which can be applied to a full sequence.

To extend the concept to a higher number of dimensions we need to apply some different ideas. Let us start to define and bound the problem in a more precise way. Assuming a sequence of states $\mathbf{x} = \{x_0, x_1, \ldots, x_T\}$, and a sequence of observations $\mathbf{y} = \{y_0, y_1, \ldots, y_T\}$, both $x_t$ and $y_t$, complying with the following equations

$$\begin{aligned} x_t &= \mathbf{A} x_{t-1} + u_t \\ y_t &= \mathbf{C} x_t + v_t \end{aligned} \tag{2.3.4}$$

representing $p(x_t|x_{t-1})$ and $p(y_t|x_t)$ and known as the state and observations equations, where $u_t$ and $v_t$ are noise sampled from a Normal distribution with mean zero and covariance respectively $\mathbf{\Gamma}$ and $\mathbf{\Sigma}$. From these equations we can see that all the distributions are Gaussians. This is the key concept and it will produce some simplifications based on the property that Gaussians are a group under multiplication.

Using a similar strategy, as in the discrete HMM case, of keeping the information regarding the past observations stored in a distribution. define

$$\alpha(x_t)\, c_t = \int p(y_t|x_t)p(x_t|x_{t-1})\alpha(x_{t-1})dx_{t-1} \tag{2.3.5}$$

as the alpha recursion. This alpha recursion is not needed directly but it will supply the equation for the relation needed to allow us calculate the values of the Kalman gain. Since all the distributions are Normal distributions we can rewrite equation (2.3.5) as

$$\alpha(x_t) = \mathcal{N}(x_t|\mu_t, V_t) = \mathcal{N}(y_t|\mathbf{C}x_t, \mathbf{\Sigma}) \int \mathcal{N}(x_t|\mathbf{A}, \mathbf{\Gamma})\mathcal{N}(x_{t-1}|\mu_{t-1}, v_{t-1})dx_{t-1} \tag{2.3.6}$$

which relates the values of the means and covariances for the previous and current states given the past observations. By solving the algebraic equations we can obtain the update equations for the states current mean and covariance,

$$\mu_t = \mathbf{A}\mu_{t-1} + \mathbf{K}_t(y_t - \mathbf{C}\mathbf{A}\mu_{t-1}) \tag{2.3.7}$$

$$\mathbf{V}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{P}_{t-1} \tag{2.3.8}$$

based on the Kalman gain $\mathbf{K}$ and auxiliary variable $\mathbf{P}$ given by

$$\mathbf{K}_t = \mathbf{P}_{t-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \mathbf{\Sigma})^{-1} \tag{2.3.9}$$

$$\mathbf{P}_{t-1} = \mathbf{A}\mathbf{V}_{t-1}\mathbf{A}^T + \mathbf{\Gamma} \tag{2.3.10}$$

and with $c_t = \mathcal{N}(y_t|\mathbf{C}\mathbf{A}\mu_{t-1}, \mathbf{C}\mathbf{P}_{t-1}\mathbf{C}^T + \mathbf{\Sigma})$. The Kalman gain is the weight affecting the difference between the forecast value for the state based on the information from the previous state and the current observation. The details and intricacies of the deduction for this equations are fully explained in the previously cited books by [Ogata 87], [Friedland 96]

and [Bishop 07].

These equations are based on the knowledge of the matrices for $\mathbf{A}, \mathbf{C}, \mathbf{\Sigma}, \mathbf{\Gamma}$ which are the parameters for Kalman filters. If the values of these parameters are unknown then we require some training data. If the training data have state and observation sequences a more rapid inference is possible. If no state sequence is present in the training data then an EM algorithm must be performed. The main idea is to generate an $\alpha$-$\beta$ recursion, as in HMM and CRF, and compute the state sequence distribution probability at each time step in a similar way to the one presented in the *Problem 3* in section 2.2.1, the details for this process can be found in [Bishop 07].

Even if there are some similarities between Kalman filters and HMM since both produce a forecast for the next state using the information from the previous state updated by the information from the observation, the concept of a Kalman filter is different from the HMM model. HMM models are a method which allow a labelling of an input, there is no formal relation between the distance, in the norm sense, of the values of the observations to the values of the labelling, Kalman filters on the other hand due to their definition are tied to the observations consequence of the linearity of the observation equation (2.3.4) and the Gaussian assumption. Kalman filters can be seen as a tracking method where the mobility of the mean, defined by the variance, forms the most probable path and in consequence performs a filtering action. The advantage of this process it is its recurrence nature which is well adapted to perform an online filtering process and in consequence is well suited for this type of applications.

Let us note that continuous state CRF which we will present later, are not comparable to Kalman filters, since continuous state CRF may not be uni-modal or even generate a Gaussian distribution, such as with Kalman filters. Also general feature functions for CRF may not be linear and they may not be separable in terms of states and observations.

## 2.4 Particle Filters

Except in the case of symbolic manipulation, some discrete support must be obtained to represent continuous distribution functions in a digital form and allow their manipulation. If an isometric distribution of Dirac-deltas, with some relevant weights, can be perceived

as a suitable form or even convenient, a little deeper thought quickly shows some serious drawbacks, since in theory we will need an infinite number of weights to be stored, if their domain is in $\mathbb{R}$. But since probability distribution functions are normalised (their integrals over their domain are equal to one) and thus bounded, we will have as a corollary that their values tend to zero as the variable goes to plus or minus infinity. Using this property we can easily see that the relevance of the points far away from the interval of interest is marginal. It is then possible to have a support with an isometric distribution of Dirac-deltas as long as we maintain a more or less accurate track of the relevant interval.

The Particle Filter (PF) main concept, as in [Doucet 01], is to carry the information of the distribution not on a set of weights linked to a known fixed position, but on a set of positions linked to known fixed weights, in this case, a weight equal to the inverse of the number of particles. If we assume the analogy, which gives PF its name, that at each position we have a particle with a given mass, we can generate a potential by adding the local potentials generated by the mass of each particle. This general potential can be normalised and as a consequence can approximate a probability density function. The advantage of this method over the previous isometric distribution becomes evident if we think that the moment we assess the relevant positions of the particles we at the same time assess the correct relevant interval.

Particle filters have several different variants, the difference lies mainly in the re-sample phase where several types of re-samples are available or even no re-sample like the Sequential Importance Sampling (SIS). Here we will restrict ourselves to Sampling Importance Re-sampling (SIR) [Doucet 01] which is the first in historic terms and so incorporates the main ideas of particle filters.

The standard method for SIR sampling for the general distribution in particle filters, is based on a forecast from a conditional probability in to two consecutive states, then there is a re-evaluation phase on the conditional probability of the observation, followed by a re-sampling.

It can be sometimes difficult to produce a notation both precise and readable or concise at the same time. For this purpose, on this work, we tried to use a notation as close as possible as the general notation of the cited references when ever it was possible. But to maintain a consistency over the sections we were forced to change notation in the aim of

readability. So, in the following, we have changed the time from subscript to superscript to maintain readability between the indexes representing a number of particles.

Let us assume we have a Markov Process of order one with

$$p(x^t|x^{t-1,\ldots,0}, \mathbf{y}) = p(x^t|x^{t-1}) \cdot p(x^t|y^t) \tag{2.4.1}$$

since it is a recursive form we have a set of particles supporting the distribution for $p(x^{t-1})$, we can sample a set of particles $q_i$ from $p(x^t|x_i^{t-1})$ and evaluate $p(q_i|y^t)$ and finally the corresponding weights for this $q_i$ sampling for $p(x^t|x^{t-1,\ldots,0}, \mathbf{y})$, a re-sample process will transform this set of weighted particles to a set of unitary weight ones supporting the $x^t$ distribution [Doucet 08].

So for the non-linear process

$$x^t = f(x^{t-1}) + u_t \qquad y^t = g(x^t) + v_t \tag{2.4.2}$$

the expectation of $f$ will be approximated by

$$\int f(x^t)p(x^t|\mathbf{y})dx^t \approx \sum_{i=0}^{N} w_i f(x_i^t) \qquad \text{where} \qquad \sum_{i}^{N} w_i = 1 \tag{2.4.3}$$

were $u_t$ and $v_t$ are noise and $w_i$ is the normalised weight of particle $i$ affecting the values of the function at given points. Given a proposal distribution $\Phi(x^t|x^{t-1}, \mathbf{y})$ which is a distribution closest to the expected distribution possible from which we plan to sample, the Sampling Importance Re-sampling becomes:

1. Sample $N$ samples from the proposal distribution $x_i^t \sim \Phi(x^t|x^{t-1}, \mathbf{y})$

2. For all the new points rescale the previous weights to the new distribution by

$$w_i^t = W_i^{t-1} \cdot \frac{p(y^t|x_i^t)p(x_i^t|x_i^{t-1})}{\Phi(x_i^t|x_i^{t-1}, \mathbf{y})} \tag{2.4.4}$$

3. Rescale the weights $w_i^t$ again so they represent a distribution with $\sum_i^N W_i = 1$.

$$W_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_j^t} \tag{2.4.5}$$

4. Calculate the effective number of particles $N_{eff} = [\sum_{i=1}^{N}(W_i^t)^2]^{-1}$

5. If the effective number of particles falls bellow a threshold $N_{eff} < N_{thrs}$ then re-sample the particles.

As we said the re-sample process has several variants, we will present here only the systematic re-sampling algorithm, which we use later. The re-sample algorithm forces the particles to maintain a range around the relevant points of the function where the values are significant and not close to zero.

1. Define $w_0^t = 0$ and sort $w_i^t$ and $x_i^t$ on $w$ in a descending order.

2. Compute the accumulated weights by $\hat{w}_i^t = w_i^t + \hat{w}_{i-1}^t$

3. Calculate the percentage of each particle for a given number of samples $a = (N+1)^{-1}$.

4. For $j$ in $\{1, \ldots, N\}$ if $\hat{w}_{i-1} < j \cdot a \leq \hat{w}_i$ then add $\{w_i^t, x_i^t\}$ to the new set of particles.

We have to stress that particle filters are not a model by themselves, but a way to support a model, they require a definition of the transition probability since they do not define one themselves. This property allows us to use it with either a Kalman filter, where particle filters have given us a different approach and perspective and allowed extensions to non-linear models [Bishop 07], or any other model with continuous distributions like a continuous state CRF as we will see in the next chapter.

## 2.5 Conclusions

We have presented in this chapter an overview of sequence classifiers such as HMMs and CRFs and extended this concept for regression with Kalman filters. We also have seen the definition of sequence classification and the Markov assumption which is used in on the three models we have presented. We have highlighted the differences between both the discrete states HMM and CRF and explained how to obtain not only the CRF formulation through Maximum Entropy but also the parameters governing HMM with the expectation maximisation algorithm (or Baum-Welch for the HMM case) and the ones for CRF with

the maximisation of the log-likelihood. A way to obtain the most probable sequence through the Viterbi algorithm was also explained.

The change in the subtle but inherent paradigm differences addressed by HMM and Kalman filters were presented, as was the Kalman filter concept as a simple one-dimensional formulation, before we briefly explained the more general case.

We have concluded this chapter by presenting particle filters which allow us to shed new light on the tracking problem addressed by Kalman filters and supply us with a different way of integrating a function composed by the product of a distribution and a more general function.

The models and concepts presented on this chapter will be used in the next chapters to extend the idea of Conditional Random Fields presented here as a discrete states model to a continuous states conditional random fields, and to try to solve the new challenges posed to us by this extension.

# Chapter 3

# Maximum Likelihood of continuous state Conditional Random Fields using Monte Carlo particles

## 3.1 Introduction

Even though we often talk about a discrete subject when we talk about digital computers, it is sometimes important in computer science to treat the modelling of real problems as continuous states problems, instead of discrete ones. In modern computers the digitisation of a continuous variable into a sixty-four bit integer will generate too large a number of admissible states to be handled by the usual discrete methods. By that, we mean that the most correct and efficient way of discretising a continuous problem may not be by a naive isometric distribution of Dirac deltas. By "isometric distribution" we mean a separation of the Dirac deltas by an equal distance, as a homothety of the natural number on the real line. Even though the capacity to hold a sufficiently large number of states and a corresponding good discrimination is available, limitations on speed and algorithms may lead to such poor performance that this strategy may not be usable, so a continuous treatment of the problem may be more profitable.

Continuous models can, in theory, handle an infinite number of states but they still need to be contained by digital computers, so they need to be discretised in some way. The difficulty is to generate a model which can be discretised in such a way that the virtues of continuity will remain untouched while the model can still be correctly handled numerically.

So the extension of Conditional Random Fields to continuous states is the logical step forward as far as the increased number of states is concerned. Continuous state Conditional Random Fields are Conditional Random Fields where the state does not belong to a finite set anymore. The extension to CRF has been made by [Taycher 06] and [Limketkai 07]. Both of them used graph arguments to obtain the relations of the transitions probabilities (or potentials) to define the log-likelihood as opposed to an extension of the deduction of [Klinger 07]. Even if, as in the case of [Taycher 06], we have a continuous observation CRF, there is an explicit reduction to a cell in a grid represented by a posture (or representative state). This classification is done with the help of a norm to assess the distance from a given state to a representative state, and thus doing an implicit discretisation, since these cells are defined a priori. In the case of [Limketkai 07] they opted for a Lafferty-type feature function which enabled them to produce a two-stage particle filter which was used to compute the Maximum A Posteriori (MAP) sequence for the trajectory of the most probable particle at the last time step for a given observation sequence. A difference vector of feature functions weights is calculated using this state sequence and the initial states sequence, which will update the weights vector. An inner loop helps to regularise the updating by weighting the gradient with an exponentially decreasing factor. Neither one of the works tried to obtain a fast learning convergence through minimizing the log-likelihood for complete continuous observation model.

We will present a different approach from the one used by [Taycher 06] and [Limketkai 07], by trying to maximise the log-likelihood through a Newton-type gradient minimizing numerical algorithm. We will closely follow the [Klinger 07] formulation for discrete CRF. So we will extend the concept of the log-likelihood through an analytical perspective in 3.3, having previously formalised the concept for continuous states in 3.2. As in [Limketkai 07] we also use particle filters for tracking, but in our case we will use them also to approximate the local potentials and as a consequence the total potential. We will finally present

our results in 3.6.

## 3.2 Maximum Entropy

To formalise the continuous states CRF through Maximum Entropy (ME) we need to define another functional beginning with the conditional entropy, but accepting this time, a continuous distribution

$$H(\mathbf{x}|\mathbf{y}) = -\int_{\mathbf{x},\mathbf{y}} p(\mathbf{x},\mathbf{y}) \log p(\mathbf{x}|\mathbf{y}) d\mathbf{y} d\mathbf{x} \tag{3.2.1}$$

this equation comes from the generalisation of the entropy formula from Shannon in [Jaynes 03]. As in the discrete case we will have the two conditions imposed by the probability axioms but this time for continuous distributions.

$$p(\mathbf{x}|\mathbf{y}) \geq 0 \qquad \text{for all } \mathbf{x},\mathbf{y} \qquad \text{and} \qquad \int_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) d\mathbf{x} = 1 \qquad \text{for all } \mathbf{y} \tag{3.2.2}$$

This entropy formulation allow us to build the new continuous Lagrangian. Where, $E(f_i)$ represents the expectation of the feature functions under the empirical distribution, which is the distribution obtain by measuring the frequency of the features in the training data. The expectation $\tilde{E}(f_i)$ is the expectation of the feature function on the general case or in the all possible outcome of sequences of states vid 2.2.2. So the new Lagrangian is

$$\Lambda(p,\lambda) = -\int_{\mathbf{x},\mathbf{y}} p(\mathbf{x},\mathbf{y}) \log p(\mathbf{x}|\mathbf{y}) d\mathbf{y} d\mathbf{x} + \sum_i \lambda_i (E(f_i) - \tilde{E}(f_i)) \tag{3.2.3}$$

which, by forcing the gradient of the functional to zero, will give us the equation of the canonical distribution:

$$p(\mathbf{x}|\mathbf{y}) = Z_\lambda(\mathbf{y}) \cdot \exp\left(\sum_i \lambda_i f_i(\mathbf{x},\mathbf{y})\right) \tag{3.2.4}$$

where $Z_\lambda(\mathbf{y})$ is a normalisation factor or "total potential". So by using the canonical distribution and by applying the second axiom, we can deduce the formula for the total potential. Let us note that the first axiom is always met because we will have a formulation

based on the exponential of a real value function. If we define $Z_\lambda(\mathbf{y})$ as

$$Z_\lambda(\mathbf{y}) = \int_{\mathbf{x}} \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, \mathbf{y})\right) d\mathbf{x} \tag{3.2.5}$$

then the complete general model is then obtained, and is given by the continuous conditional likelihood which is very similar to the discrete formulation but we have transformed the summation over the set of states, into an integration over the space of continuous observations.

## 3.3 Maximum Likelihood

We can now calculate the likelihood after first checking we have a consistent formulation. To check this consistency, we need to define the feature function's constraints. The only constraint we need to impose on the feature functions is their integrability, so they need to comply with

$$\int_\Omega |f_i(s, v, \mathbf{y}, t)| ds dv < M \quad \text{with} \quad M \in \mathbb{R}, \forall i, \mathbf{y}, t \tag{3.3.1}$$

Strictly speaking the domain of the functions does not need to be restricted to real-value states or observations, however it is convenient from a manipulation point of view to do so. This constraint is only marginally relevant in terms of applications since all the states and observations are, in reality, real values, so $x^t \in \mathbb{R}$ which implies $\Omega = \mathbb{R}^{2T}$, where $T$ is the number of observations. On the other hand it is important that the co-domain be defined in $\mathbb{R}$ to maintain the stability of the exponential.

Assuming we have $I$ feature functions, we can express the new likelihood for the continuous states model as

$$\mathcal{L} = p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{\exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right)}{\int_{\mathbb{R}^T} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) d\mathbf{x}} \tag{3.3.2}$$

and log likelihood

$$\log(\mathcal{L}) = \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)}_{A} - \underbrace{\log\left(\int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) d\mathbf{x}\right)}_{B} \tag{3.3.3}$$

In the discrete case, we evaluate the expectation by analysing all the possible outcomes for $p(\mathbf{x}|\mathbf{y})$ by doing a summation on all possible states, after the generalisation to continuous observations, this is equivalent to integrating in the $\mathbb{R}^T$ space.

The difficulty here is to perform an integral in high-dimensional space. Integrals in high dimensions spaces are notoriously difficult, this is known as the "curse of dimensionality" [Evans 00]. General methods like the simple Riemann integration will require so many evaluation points for the integrating functions that they become quickly unsuitable as the dimensions increase. The formal definition and relevant theorems for the Riemann integration can be found in [Priestley 97]. Other numerical algorithms which work very well in low dimensions start to loose their efficiency as the number of dimensions gets bigger. There are some integration methods which can cope with high dimensional integrals and they are closely related to Monte Carlo methods; for example the Markov Chain Monte Carlo or Particle Filters [Evans 00]. In our case we have a sequence of interconnected integrals which may be suitable for the application of Particle Filters.

As discussed in the previous chapter we will need to use a numerical method to maximise the log-likelihood, so one needs to find the normalization factor $Z_\lambda(\mathbf{y})$. To improve the convergence rate by using a more efficient algorithm, we can analytically calculate the gradient of the log-likelihood by mathematical manipulation.

So by re-writing the normalisation constant we have,

$$Z_\lambda(\mathbf{y}) = \int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) d\mathbf{x} \tag{3.3.4}$$

and if we expand the integral into its individual components,

$$Z_\lambda(\mathbf{y}) = \int \dots \int \int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) dx^0 dx^1 \dots dx^T \tag{3.3.5}$$

we can now, using the commutativity property of multiplications, separate the first factor inside the integral to obtain

$$Z_\lambda(\mathbf{y}) = \int \dots \int \int \exp\left(\sum_{t=2}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) \cdot \exp\left(\sum_{i=1}^{I} \lambda_i f_i(x^0, x^1, \mathbf{y}, t)\right) dx^0 dx^1 \dots dx^T \tag{3.3.6}$$

since the other factors are independent of the first factor, we can then integrate it, separately, since the others can be considered as constants for this single integration.

$$Z_\lambda(\mathbf{y}) = \int \dots \int \exp\left(\sum_{t=2}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) \cdot \underbrace{\left[\int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(x^0, x^1, \mathbf{y}, t)\right) dx^0\right]}_{\alpha^0(x^1)} dx^1 \dots dx^T \tag{3.3.7}$$

We can not extract this integral from the rest of the integration because of the integration in $x^1$ but we can transform this single integration into a function. So by defining the integral of the first factor as

$$\alpha^0(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(x^0, v, \mathbf{y}, t)\right) dx^0 \tag{3.3.8}$$

the scale factor or total potential becomes

$$Z_\lambda(\mathbf{y}) = \int \dots \int \exp\left(\sum_{t=2}^{T}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) \cdot \alpha^0(x^1) dx^1 \dots dx^T \tag{3.3.9}$$

Using a sequence of integrals which at each time only depends on one factor and the previous element of the sequence,

$$\alpha^t(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t)\right) \cdot \alpha^{t-1}(s) ds \tag{3.3.10}$$

we can re-write the scale factor as

$$Z_\lambda(\mathbf{y}) = \int \alpha^T(v)dv \tag{3.3.11}$$

This is equivalent to an $\alpha$ recursion of the discrete CRF or HMM. It is clear that the same sort of arguments can be used to produce a $\beta$ recursion. So for the $\beta$ recursion we have;

$$Z_\lambda(\mathbf{y}) = \int \dots \int \underbrace{\left[\int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(x^{T-1}, x^T, \mathbf{y}, t)\right) dx^T\right]}_{\beta_T(x^T)} \cdot \exp\left(\sum_{t=1}^{T-1}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) dx^0 \dots dx^{T-1} \tag{3.3.12}$$

or

$$Z_\lambda(\mathbf{y}) = \int \dots \int \beta_T(y^T) \cdot \exp\left(\sum_{t=1}^{T-1}\sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) dx^0 \dots dx^{T-1} \tag{3.3.13}$$

with

$$\beta^t(s) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t)\right) \cdot \beta^{t+1}(v)dv \tag{3.3.14}$$

and

$$\beta^T(s) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t)\right) \cdot dv \tag{3.3.15}$$

Of course there is no need to compute the total potential twice, but we will need a $\beta$ recursion later on to assess the derivative of the log-likelihood by calculating the expectation of each feature function.

To present this formulation in a more succinct way, and let us remember we defined the local potential as $\Psi^t(s, v, \mathbf{y}, t) = \exp(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t))$ or $\Psi^t(s, v)$ for a short notation, we can then write the recurrence as

$$\alpha^t(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t)\right) \cdot \alpha^{t-1}(s)ds = \int \Psi^t(s, v) \cdot \alpha^{t-1}(s)ds \tag{3.3.16}$$

or for the $\beta$ expression

$$\beta^t(s) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{y}, t)\right) \cdot \beta^{t+1}(v)dv = \int \Psi^t(s, v) \cdot \beta^{t+1}(v)dv \tag{3.3.17}$$

So we have transformed an integration over a high-dimensional space in to a sequence of recurrent integrals. For this integration we need now to give numerical support for these continuous distributions. The chosen approach was then the use of Particles Filter to approximate those integrals. The difficulty here is the generality of the feature functions, since we have assumed any type of feature functions are acceptable, so a [Klinger 07] type function in opposition to a [Lafferty 01] type which assume they are two disjoint probabilities and are split in two separable "families" of functions, one proportional to the transition probability given the previous states $p(x^t|x^{t-1,\ldots,0})$, and one proportional to the observation probability given the current state $p(\mathbf{y}|x^t, t)$. It is not clear that the second type of feature functions ([Lafferty 01] type) spans the same space as the [Klinger 07] type.

## 3.4 Maximum Likelihood with Monte Carlo Particles

Based on the equations deduced in the previous section and the Particle Filters concepts presented on the previous chapter it is now possible to deduce the equations governing the calculation of the addictive inverse of the log-likelihood using the $\alpha$ recursion and the gradient of this function using a $\alpha$-$\beta$ recursion using particle filters to produce the integrations. This will be done by using the particle filters with importance sampling with a proposed density chosen to be a mixture of Gaussians centered on the particles from the previous time step.

The usual Particle Filter approach is to use the forecast from the transition probability $p(x^t|x^{t-1}, \ldots, x^0)$ as a sampling distribution of the position of the particles in the next time step, then evaluate this forecast using the observation probability $p(y^t|x^t)$ to re-sample the particles in the approximated distribution in the time step $t + 1$. In our case only a total evaluation is possible through the potential which is proportional to the transition probability given the observation or $\Psi^t(x^{t-1}, x^t, \mathbf{y}, t) \propto p(x^t|x^{t-1}, \mathbf{y}, t)$.

This forces us to redefine a different sampling method from the most popular Particle Filter sampling methods, since [Klinger 07] type continuous observation feature functions impose a different paradigm and we must try to overcome this difficulty.

If we assume a sufficient small time step and a well behaved distribution with a bounded second derivative, then we can assume that the transition probability can be approximated

by a Gaussian. With this assumption we can then generate a forecast distribution in the next time step. Having a forecast of the particles positions for the next time step, allows us to properly calculate their probability, or potential, in the case of CRF. So our proposed position for a given particle and for the next time step is obtained by the current location plus a Gaussian perturbation.

We have seen in equation (2.4.3) that if $p(u)$ has the properties of a probability distribution and if $u_m \sim p(u)$, then

$$\int p(u)f(u)du = \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} f(u_m) \tag{3.4.1}$$

The potentials $\Psi^t(s, v)$ have all the properties of a probability distribution except for the unitary condition. In fact $\int \alpha^t(v)dv$ may not be one. On the other hand $\Phi^t(v) = \frac{1}{Z^t}\alpha^t(v)$ where $Z^t = \int \alpha^t(v) \cdot dv$ has all the properties of a probability distribution.

Let us calculate the local potentials for a given time step $t$. For this purpose we define a new probability distribution $\Phi$, by generating noise around a set of points sampled from the previous distribution, or $u_i \sim \Phi^{t-1}(v)$, so the proposal distribution will be

$$q(s) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(u_i, \sigma^2 I) \tag{3.4.2}$$

The proposal distribution is an arbitrary distribution expected to be close to the distribution to be approximated. We can rewrite the local potential ${}_\alpha Z^t$ as

$${}_\alpha Z^t = \int \alpha^t(v) \cdot dv = \int \frac{\alpha^t(v)}{q(v)} \cdot q(v) \cdot dv \tag{3.4.3}$$

so by equation (3.4.1) where we sample s set of $M$ points from the proposal distribution $p_j \sim q(s)$.

$${}_\alpha Z^t \approx \frac{1}{M} \sum_{i=1}^{M} \frac{\alpha(p_j)}{q(p_j)} \tag{3.4.4}$$

The difficulty in this case is to calculate $\alpha(p_j)$ because we need to sample from it, but let us remember again that

$$\alpha^t(v) = {}_\alpha Z^{t-1} \int \Phi^{t-1}(s)\Psi^t(s, v) \cdot ds \tag{3.4.5}$$

since $\alpha^{t-1}(s) = {}_{\alpha}Z^{t-1}\Phi^{t-1}(s)$ and $u_i$ are sampled from the distribution $\Phi^{t-1}(v)$ again by the integration formula (3.4.1) we end up with

$$\alpha^t(v) = {}_{\alpha}Z^{t-1}\frac{1}{N}\sum_{i=1}^{N}\Psi(u_i, v) \tag{3.4.6}$$

So to calculate the local potential at time $t$ we have

$${}_{\alpha}Z^t = \int \alpha^t(v)dv = \int {}_{\alpha}Z^{t-1}\frac{1}{N}\sum_{i=1}^{N}\Psi(u_i, v)\cdot dv \tag{3.4.7}$$

and now we can apply equation (3.4.4) to obtain the final formula for the local potential.

$${}_{\alpha}Z^t = {}_{\alpha}Z^{t-1}\frac{1}{N\cdot M}\sum_{j=1}^{M}\sum_{i=1}^{N}\frac{\Psi^t(u_i, p_j)}{q(p_j)} \tag{3.4.8}$$

Let us now compute the samples $u_i^t$ from the distribution $\Phi^t(s)$. For that, let us assume a set of weights affecting each $p_j$ given by

$$w_j = \sum_{i=1}^{N}\Psi(u_i, p_j) \tag{3.4.9}$$

a new set of unitary weight $W_i$ with the property of $\sum W_i = 1$

$$W_j = \frac{w_j}{\sum_{i=1}^{N} w_i} \tag{3.4.10}$$

now we can re-sample $p_j$ to obtain a new sample $u_i^t$ supporting $\Phi^t$

$$u_i^t \sim \text{systematic re-sample}(p_j, W_i) \tag{3.4.11}$$

where the systematic re-sample is the Particle Filter systematic re-sample scheme explained in part 2.4.

Using a Gaussian, as a source of noise or a proposal density for the perturbation of the particles, can be seen as a diffusion of the particles. Implicitly it imposes constraints on the ability of the model to some extent. A quick and fast-changing system may be hard to track with few particles and small standard deviation on the Gaussian. On the other hand a big standard deviation may lead to poor performance in terms of accuracy. The

accuracy might be degraded because a larger number of particles will not be representative since they will lie on the lower values of the function.

## 3.5 Gradient of the Maximum Likelihood with Monte Carlo Particles

In order to use a Newton-type efficient algorithm for maximising the log-likelihood as in the Limited Memory Variable Metrics (LMVM) [Benson 01], we need to compute the gradient of the function to minimise. The the gradient of part $A$ of equation 3.3.3, is given by

$$\frac{\partial A}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l} \sum_{t=1}^{T} \sum_{i=1}^{I} \lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t) = \sum_{t=1}^{T} f_l(x^{t-1}, x^t, \mathbf{y}, t) \tag{3.5.1}$$

as in the discrete case, this corresponds to the expectation of the feature functions $E(f_l)$ of the training data and there is no change in the expression.

As for part $B$, the formula is more complex. We first differentiate with respect to the log functions

$$\frac{\partial B}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l} \log Z_\lambda(\mathbf{y}) = \frac{1}{Z_\lambda(\mathbf{y})} \frac{\partial Z_\lambda(\mathbf{y})}{\partial \lambda_l} \tag{3.5.2}$$

then expand the $Z_\lambda(\mathbf{y})$ factor for the partial derivative part

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \frac{\partial}{\partial \lambda_l} \int_{\mathbb{R}^T} \exp\left( \sum_{t=1}^{T} \sum_{i=1}^{I} \lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t) \right) d\mathbf{s} \tag{3.5.3}$$

since the derivative variables are different from the integrating ones, it is possible to differentiate inside the integral

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \int_{\mathbb{R}^T} \exp\left( \sum_{t=1}^{T} \sum_{i=1}^{I} \lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t) \right) \cdot \sum_{t'=1}^{T} f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t)\, d\mathbf{s} \tag{3.5.4}$$

We can easily see in the previous equation the same principle as in the discrete case; we are calculating the expectation over all possible state sequences of each feature function $f_l$ of all the possible sequences. However this time, the number of sequences, instead of being finite is now infinite [Priestley 97].

Taking the previous equation and, due to the distributive law of the group since we are working in $\mathbb{R}$, we re-arrange the equation so the multiplying factor affecting the total

summation will instead multiply each element of the summation

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \int_{\mathbb{R}^T} \sum_{t'=1}^{T} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(s^t, s^t, \mathbf{y}, t)\right) \cdot f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t)\, d\mathbf{s} \tag{3.5.5}$$

The integral operator having the property of linearity, and thus additivity, we can now extract the summation outside of the integral, and we end up with

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \sum_{t'=1}^{T} \int_{\mathbb{R}^T} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t)\right) \cdot f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t)\, d\mathbf{s} \tag{3.5.6}$$

By analysing the expression (3.5.6) we can see a summation of integrals which, in each one, a factor affects only an element in a given $t$. So we can use the property of independence as we did previously to calculate a recurrent sequence for $B$. To do so let us reformulate the expression (3.3.11) which gives us $B$

$$B = \int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I} \lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t)\right) d\mathbf{s} \tag{3.5.7}$$

Transforming the summation inside the exponential into a product

$$B = \int \prod_{t=1}^{T} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t)\right) d\mathbf{s} \tag{3.5.8}$$

and using the potentials $\Psi$ instead of the exponential to simplify the notation

$$B = \int \prod_{t=1}^{T} \Psi^t(s^{t-1}, s^t) \cdot d\mathbf{s} \tag{3.5.9}$$

we can expand the integral in to its components

$$B = \int \int \ldots \int \prod_{t=1}^{T} \Psi^t(s^{t-1}, s^t) \cdot ds^T \ldots ds^1 \cdot ds^0 \tag{3.5.10}$$

Now, expanding the product in a sequence of three products; the first composed of the products of the potentials from the beginning of the sequence at time 1 to $t-1$, the second as the potential for the transition between $t-1$ and $t$, and the third the products of the

potential from $t+1$ to the end of the sequence at time $T$.

$$B = \int \dots \int \left[ \prod_{t'=1}^{t-1} \Psi^{t'}(s^{t'-1}, s^{t'}) \right] \cdot \Psi^{t}(s^{t-1}, s^{t}) \cdot \left[ \prod_{t''=t}^{T} \Psi^{t''}(s^{t''-1}, s^{t''}) \right] \cdot ds^{T} \dots ds^{t+1} ds^{t-2} \dots ds^{0} \cdot ds^{t} ds^{t-1} \tag{3.5.11}$$

Using once more the same argument of independence of factors not in the immediate vicinity to aggregate the integral in each independent factor, we have:

$$B = \int \int \left[ \int \dots \prod_{t'=1}^{t-1} \Psi^{t'}(s^{t'-1}, s^{t'}) ds^{t-2} \dots ds^{0} \right] \cdot \Psi^{t}(s^{t-1}, s^{t}) \cdot \left[ \int \dots \prod_{t''=t+1}^{T} \Psi^{t''}(s^{t''-1}, s^{t''}) ds^{T} \dots ds^{t+1} \right] \cdot ds^{t} ds^{t-1} \tag{3.5.12}$$

and finally using the $\alpha$ and $\beta$ functions to represent the partial integrals.

$$B = \int \int \alpha(s^{t-1}) \cdot \Psi(s^{t-1}, s^{t}) \cdot \beta(s^{t}) \cdot ds^{t}\, ds^{t-1} \tag{3.5.13}$$

Applying the same principle we just explained to the gradient expression (3.5.6) and using the same arguments again we obtain

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \sum_{t'=1}^{T} \int_{\mathbb{R}^T} \prod_{t=1}^{T} \Psi^{t}(s^{t-1}, s^{t}) \cdot f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t)\, ds \tag{3.5.14}$$

Again splitting the product but now at time $t'$

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \sum_{t'=1}^{T} \int_{\mathbb{R}^T} \left[ \prod_{\bar{t}=1}^{t'-1} \Psi^{\bar{t}}(s^{\bar{t}-1}, s^{\bar{t}}) \right] \cdot \Psi^{t'}(s^{t'-1}, s^{t'}) \cdot f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t) \left[ \prod_{\bar{\bar{t}}=t'+1}^{T} \Psi^{\bar{\bar{t}}}(s^{\bar{\bar{t}}-1}, s^{\bar{\bar{t}}}) \right] ds \tag{3.5.15}$$

using the same argument as previously of integrating independently each product since the products are independent of the variables of $f_l$ we have

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \sum_{t=1}^{T} \int \int \alpha^{t-1} \cdot \Psi^{t}(s^{t-1}, s^{t}) \cdot f_l(s^{t-1}, s^{t}, \mathbf{y}, t) \cdot \beta^{t}\, ds^{t} ds^{t-1} \tag{3.5.16}$$

So the re-arranging of the equation by algebraic manipulations using the multiplication distributive property allow us to find the equation governing the calculation of the negative log-likelihood and its gradient. Now we will use Monte Carlo particles to produce the equation in a form which allows us to produce an algorithm. The term Monte Carlo particles is used instead of Particle Filter because in particle filters there is a forecast step and an updating step defined by the model and in the CRF case there is only a updating step. Since the forecast step is not in the model but is consequence of the assumption that the forecast is produced by a mixture of Gaussians centered on the particle at the previous time step. Also in particle filters the assessment is based on probabilities and in this case it is based on the local potentials. So after the approximation of the integral by the Monte Carlo Particles method we have

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{y})} \sum_{t'=1}^{T} \int\int {}_\alpha Z^{t'-1} {}_\alpha \Phi^{t'-1} \cdot \Psi^{t'}(s^{t'-1}, s^{t'}) \cdot f_l(s^{t'-1}, s^{t'}, \mathbf{y}, t') \cdot {}_\beta Z^{t'} {}_\beta \Phi^{t'} \, ds^{t'} ds^{t'-1} \tag{3.5.17}$$

collecting ${}_\alpha Z^{t-1}$, ${}_\beta Z^{t}$ and $Z_\lambda(\mathbf{y})$ into a single entity

$$\frac{\partial B}{\partial \lambda_l} = \sum_{t=1}^{T} \frac{{}_\alpha Z^{t-1} \cdot {}_\beta Z^{t}}{Z_\lambda(\mathbf{y})} \int\int {}_\alpha \Phi^{t-1} \cdot \Psi^{t}(s^{t-1}, s^{t}) \cdot f_l(s^{t-1}, s^{t}, \mathbf{y}, t) \cdot {}_\beta \Phi^{t} \, ds^{t} ds^{t-1} \tag{3.5.18}$$

and by sampling from the two distributions ${}_\alpha \Phi^{t-1}$ and ${}_\beta \Phi^{t}$ represent the alpha and beta functions at each time step divided by its integrals

$${}_\alpha u_i^{t'-1} \sim {}_\alpha \Phi^{t'-1} \qquad {}_\beta u_j^{t'} \sim {}_\beta \Phi^{t'} \tag{3.5.19}$$

we end up with

$$\frac{\partial B}{\partial \lambda_l} \approx \sum_{t=1}^{T} \frac{{}_\alpha Z^{t-1} \cdot {}_\beta Z^{t}}{Z_\lambda(\mathbf{y})} \sum_{i=1}^{N} \sum_{j=1}^{N} \Psi^{t}({}_\alpha u_i^{t-1}, {}_\beta u_j^{t}) \cdot f_l({}_\alpha u_i^{t-1}, {}_\beta u_j^{t}, \mathbf{y}, t) \tag{3.5.20}$$

which is the expression giving the gradient of $B$ in relation to $\lambda_l$ using the previous calculated ${}_\alpha Z^{t}, {}_\beta Z^{t}, {}_\alpha u_i^{t}, {}_\beta u_i^{t}$ calculated from an $\alpha$-$\beta$ recursion type of implementation.

Here, as in the HMM case, an overflow problem may occur in the $\alpha$-$\beta$ recursion calculations. But for this case it is possible to use a simpler operation by only storing the product of the local potentials ${}_\alpha Z^{t} \; {}_\beta Z^{t}$ under a logarithm form.

As in the case presented in the first chapter when dealing with the HMM model there is also the issue of numerical stability. But here we do not need to "unload" the $\alpha$ or $\beta$ values since we have the integrals values and a distribution stored on the weight of the particles. There is only the need to store the local potentials under a logarithmic form.

We have thus obtained the equations governing the log-likelihood at its gradient under a Monte Carlo particles framework which allow us to produce an algorithm tractable under a discrete machine like computers.

## 3.6 Results

To test our proposed approach for continuous state CRF, we have set up a mixed test problem. This test problem is based on a continuous states function but with a discrete variation. We have settled for a simple, continuous, smooth and periodic function for the states, and a quadratic for the observations as stated in (3.6.1).

$$\begin{aligned} x^t &= \sin(t/4.0) \\ y^t &= (x^t)^2 \end{aligned} \tag{3.6.1}$$

The combination of these two functions generates a "hard" non-linear relation between observations and states of time. This is because the quadratic function filters the information regarding the sign value of the states, so the observations are unable to "see" the sign of the states. For that reason we needed to artificially feed this information through the feature functions. Another option would be to generate a trigger or square signal as a new observation sequence to carry this information in a new dimensions on the observation sequence. This implies implicitly the need for the model to discriminate between two states at the same time; to follow a signal and filter the noise. In our opinion even if the problems *per se* are very simple, their combination may not be as simple.

The next step is to define the feature functions for this problem. We have summarised the feature functions in Table 3.1. Feature function $f_0$ is for tracking purposes for time $t$. The $f_1$ feature function does the filtering by imposing a probabilistic bound on the derivative and as in consequence generating a low-pass filter. The last feature function will switch the system on the most probable half-plane taking account of the sign non-

| $f_i = f(s, v, \mathbf{y}, t)$ |
|---|
| $f_0 = -(v^2 - y^t)^2$ |
| $f_1 = -(s - v)^2$ |
| $f_2 = \begin{cases} 1 & \text{if } \text{sign}(v) = \text{sign}(\sin(t/4.0)) \\ 0 & \text{otherwise} \end{cases}$ |

Table 3.1: Set #1 of feature functions for testing.

| $f_i = f(s, v, \mathbf{y}, t)$ |
|---|
| $f_0 = -(v^2 - y_t)^2$ |
| $f_1 = -(s^2 - y_{t-1})^2$ |
| $f_2 = -(s - v)^2$ |
| $f_3 = \begin{cases} 1 & \text{if } \text{sign}(v) = \text{sign}(\sin(t/4.0)) \\ 0 & \text{otherwise} \end{cases}$ |
| $f_4 = \begin{cases} 1 & \text{if } \text{sign}(s) = \text{sign}(\sin((t-1)/4.0)) \\ 0 & \text{otherwise} \end{cases}$ |

Table 3.2: Set # 2 of feature functions for testing.

linearity.

We tested a new second set of feature functions, presented in Table 3.2. In this case functions $f_2$ and $f_4$ are not an accepted relation of continuous HMM since they depend on $y_{t-1}$ and therefore breach the Markov property. By having two concurrent feature functions there is the possibility of this helping to correctly track the states based on the observations.

Particle filters were design to track signals with noise. In our case we need to make sure the particles track properly since even if there is no noise the sign change may cause difficulties. So there is the need to test the tracking behavior of the particles. The results presented here were produced by an example with no noise either on the states or on the observations and we maintained the values of all the $\lambda$ constants equal to the arbitrary value of 10, except for $\lambda_0$.

In Figures 3.1 and 3.2 we show the alpha and beta values of each particle in the vertical axis in relation to time plotted on the horizontal axis for two different values of $\lambda_0$ there is a difference in the tracking given different values of $\lambda_0$. The values for the standard deviation of the Gaussian noise of the prediction process was 0.5, fifty particles

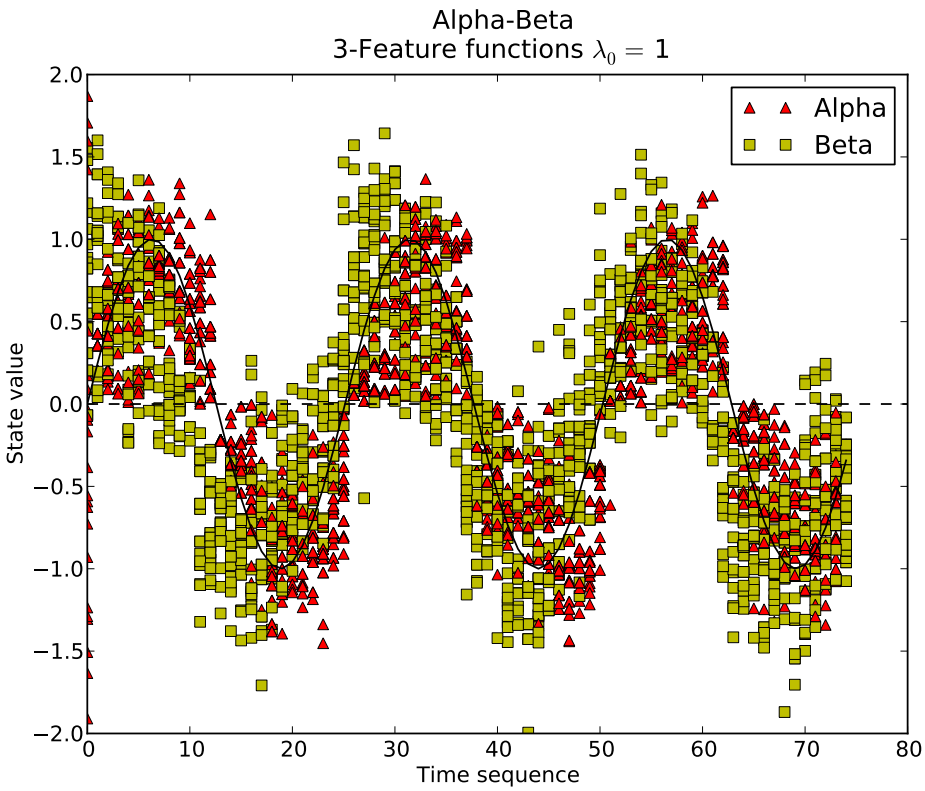


Figure 3.1: #1 set of feature functions with $\lambda_0 = 1$.

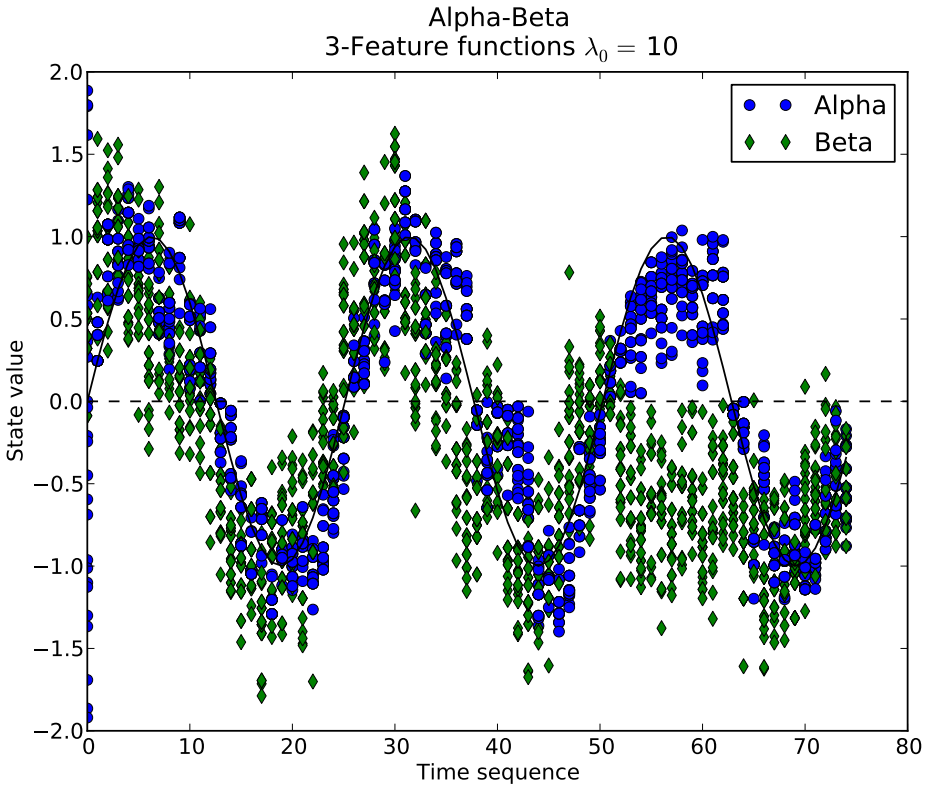


Figure 3.2: #1 set of feature functions with $\lambda_0 = 10$.

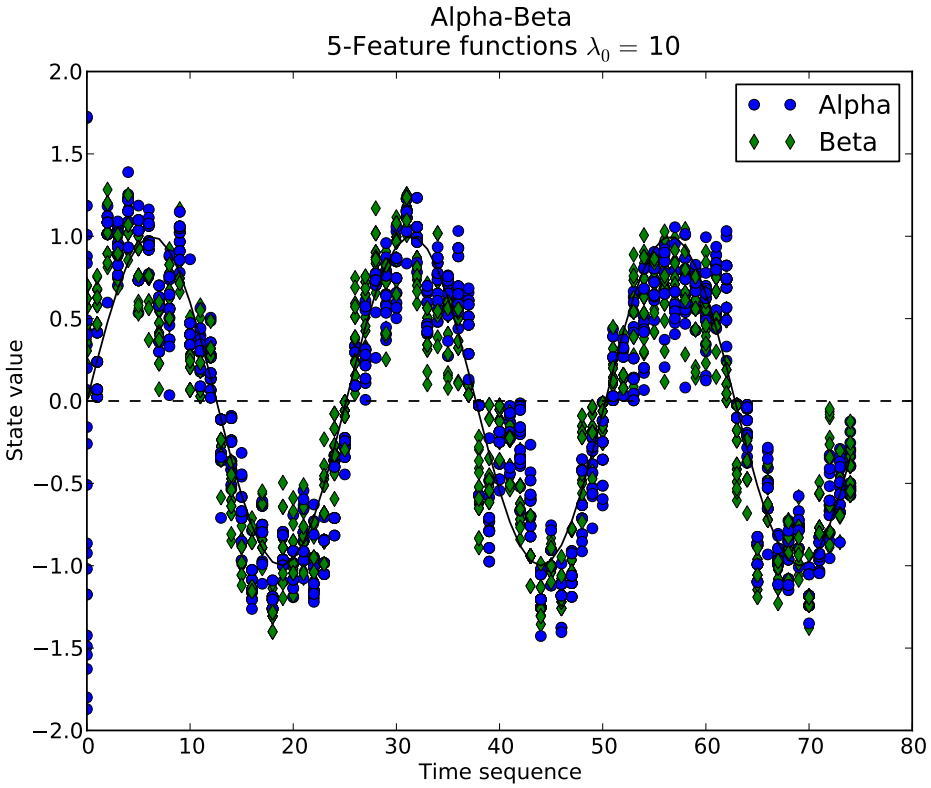


Figure 3.3: #2 set of feature functions with $\lambda_0 = 10$.

were used, initialized at a zero value. So from these figures it is possible to see there is a difference of tracking for different values of $\lambda_0$, both of them track quite well. This is not entirely unexpected since the weight of the tracking feature function, which is closely correlated with the probability, has been changed. Also as expected, as the value of the $\lambda$ for that feature function increases, so too does the efficiency of the tracking; we can see the particles are less spread. This is because the probability affecting this feature function in comparison with the remaining feature functions has increased. In both figures we can see a slight delay in the tracking, the particles are more to the right for the alpha recursion and more to the left for the beta. This is consequence of the diffusion process devised to obtain a two step Particle Filter iteration process based on a [Klinger 07] type of feature function as we explained in the previous section. The Gaussian used to perform the forecast step on the particle filter process acting on the particles is equivalent to a diffusion process.

An important point in Figure 3.2 is the tracking error for the backward pass. If the value for $\lambda_2$ is increased then the error might be reduce and in consequence reduce the

problem of unprecise tracking. Even with an improved weight for the tracking feature function there is always a possibility of a tracking error. This error is a consequence of the statistical nature of the particle filter but also a consequence of the model and the problem. Without the $f_3$ feature function the problem is bi-modal since it is composed of two Gaussian distributions so there is a probability for the model to track the lobe with the incorrect sign. Or in other words continuous state CRFs can not distinguish between the two Gaussian centred in the state and in its negation.

If we now compare Figure 3.2 and Figure 3.3 we can see a more precise tracking since we have increased the relative weight of the tracking feature function by increasing $\lambda_0$. Let us remember that we are only varying one of the weights in the $\lambda$ vector, so the tracking mechanism has more relative weight, since now we have two feature functions with the same weight performing the same work.

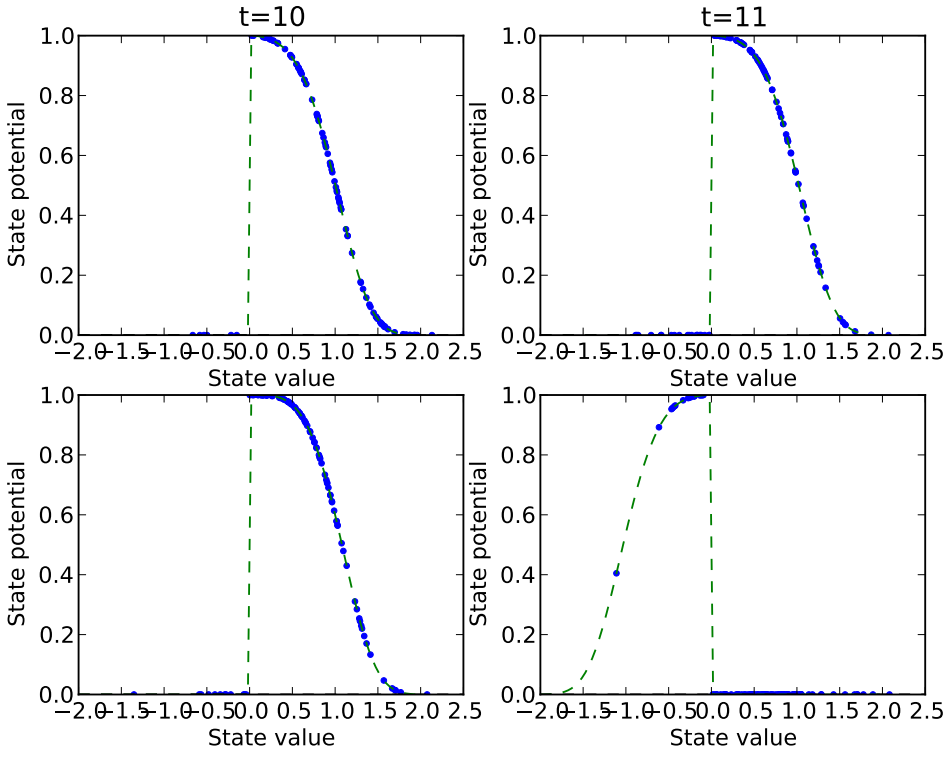


Figure 3.4: Alpha and Beta pass re-normalised potentials from Riemann and Particle Filter integration for $t = 10, 11$.

To further verify the behaviour of the tracking we have re-calculated the problem under the same conditions under a Riemann integration scheme for comparison pur-

poses. We assessed the potential of a transition with a Particle Filter scheme and a Riemann integration. Using the equations obtained in section 3.3 it was easy to obtain a formulation using Riemann integration. We assume the Riemann integration to be $\int x\,dx = \lim_{n\to\infty} \sum_{i=0}^{n} f(x_i)\Delta x_i$, and the equations for CRF with this integration can be found in section 4.3. In fact this formulation is very similar to discrete state CRFs. To maintain consistency we have used the same parameters as in the particle filters case except for the number of particles because we now used five hundred intervals. We have plotted in Figure 3.4 $\alpha$ and $\beta$ potential for two time steps. The Particle Filter integration values are represented by dots and the Riemann integration by a dashed line. To plot this graph we had to re-normalise the transition potential function $\psi^t(s, v, \mathbf{y}, \lambda)$ based on the maximum of all of the transition potentials, the calculated particles for Particle Filters and points for the Riemann integration. All the attempts to re-normalise it into a probability by using the local potentials were unsuccessful since none of them managed to match the same degree of fit as in the re-normalisation with the maximum. This leads us to think that the problem of lack of convergence of the function in the minimizing algorithm lay with the integration process. Since by re-normalising with the maximum a good fit between the $\alpha$ and $\beta$ functions is obtained contrary to a renormalization with the potential the conclusion must be the local potential present problem on its calculation, and the integration of a alpha or beta functions for a given time is the local potential. This forces us to conclude it is the integration which is responsible for the miscalculation of the $\alpha$ and $\beta$ functions.

Extending the verification of our hypothesis we have set a new comparison between the integration using the particle filter scheme and a Riemann type integration. We compared the value of the negative log-likelihood obtained by the two methods and also the partial derivative of this error function regarding the variation of the parameter $\lambda_0$ affecting the first feature function. We present the results in the Figure 3.5. Let's remember that this figure is only implicitly related with the problems we have defined previously. This figure is the result of the comparison of two integration processes so the data and the feature functions remains constant, so this figure does not directly relate to the the particular problems we defined previously.

The precision of the integration of all the Monte Carlo type methods is based on the

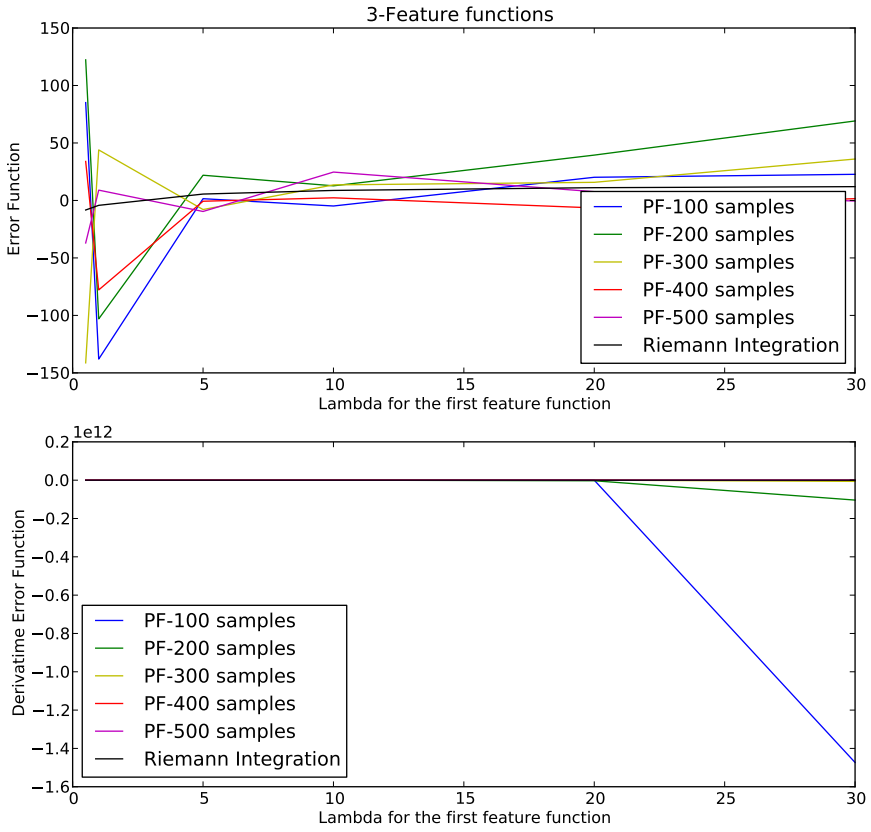


Figure 3.5: Comparison between PF and Riemann integration for a range of $\lambda_1$.

number of particles, so this precision will increase until it reaches a "saturation point", which is produced by numerical rounding errors. So we have tested the computation of the error function and its derivative with several sets of particles with different sizes.

As expected, as we increase the number of particles, the values of the error function will converge to the values of the Riemann integration scheme. Nevertheless, the error in the particle filter scheme is obvious and it is further amplified in the gradient terms. So everything indicates that if we used a sufficiently large number of particles we will reach a precision which allows us to correctly assess the local potentials and, as a consequence, the total potential. But let us remember that for each evaluation of the potential, at each point we need to compute the value of $\psi^t(s, v, \mathbf{y}, \lambda)$ for each previous particle, so the computation time will increase polynomially with the number of particles and linearly with time. So the time for convergence will have to be affected by these values. It is a well known fact that the convergence of Monte Carlo processes are very slow and the

precision converges at the rate of $\sqrt{n}$ where $n$ is the number of particles [Evans 00]. So the computation time will increase in an unacceptable way before it reaches a precision high enough for the convergence of the optimization algorithm.

Let us remember that the evaluation time for the same forward and backward recursion given a number of particles in the case of particle filters, and the same number of intervals for the Riemann integrations are in theory equivalent. The main difference lies in the rate of convergence of each integration method.

## 3.7 Conclusions

To evaluate the performance of continuous states CRFs the good tracking behaviour of the Monte Carlo particles was confirmed. While verifying the calculations of the $\alpha$ and $\beta$ functions using Particle Filters, comparing them with $\alpha$ and $\beta$ obtained with a Riemann integration with a large number of intervals, a discrepancy was noticed. By plotting these functions it was possible to conclude the reason for this discrepancy was the integration process since the one obtained by Particle Filters and by a Riemann integration did not match, contrary to the re-normalised functions with the maximum value of each function. To further confirm this evidence a test varying the number of Monte Carlo particles was performed. The results showed better performance for the ones obtained with higher number of particles compared with the results obtained with the Riemann integration.

As we have presented in the previous section, the particle filters method does not allow us to correctly calculate the factors of the feature functions which define the maximum of the log-likelihood. The main reason for this fact is the slow convergence of the particle filter integrand formula for the number of particles.

To correctly assess the expectation of each feature function, which is the core element for the calculation of the gradient of the error function, one needs to correctly compute the total potential and the potentials of the left and right sequence at the considered time step. By obtaining an error that is too large at each integration of each individual potential, we are obtaining an excessive error, for which the maximum bound is cumulative, on calculating the local potential given the rest of the sequence.

The models require, at each evaluation of a given particle at time $t$, the additional cal-

culation of the transition potential from all the particles in time $t-1$. So when we increase the number of particles, the time for evaluating the each time step will be dramatically increased also. Even with small time sequences and a reasonable number of particles, the error does not allow the LMVM algorithm to converge.

Two more factors come in to play to complicate the convergence to the minimum. By using a stochastic method as opposed to a deterministic one, at each evaluation we have an error associated with the distribution of the particles. So, at each iteration of the optimization algorithm, the function changes its shape. Even with the same parameters, and since we are doing a sampling, the values of the error function will not be the same. The dimension of the change is again based both on the number of particles and implicitly with the ability of correctly calculating the local potentials and the total potential. The second difficulty is the finite probability of losing track of the states, there is no confirmation that, at any iteration, the particles follow the correct path of the states. In the case of this event, the deformation of the potential will be even more important and biasing the information given to the optimization algorithm.

So everything indicates that in order to correctly train a continuous state Conditional Random Field, the requirements are: a deterministic method which prevents the change in shape of the error function at each iteration, gives a minimum error in the integration of the local potentials and consequently a minimal error on the total potential, with the minimum number of evaluating points to maintain a feasible computation time.

# Chapter 4

# Using Adaptive Integration for Learning Conditional Random Fields

## 4.1 Introduction

It has been seen in the previous chapter, that to correctly train a continuous state CRF one needs to precisely calculate a sequence of integrals. The need for good approximation of a integral with the least computational cost is not only a CRF problem, it extends to other problems as well. Calculating integrals by numerical methods is a well studied field for low dimensions so several different methods are available like the Newton-Cote rule or the Gauss rule quadratures [Evans 00]. But in our case there is one issue which causes difficulties: there is the need to calculate a sequence of one dimensional integrals present in the alpha beta functions evaluations and in the gradient of the negative log-likelihood calculation, but which are bound together by one of the variables. This is equivalent to an integral in a higher-dimensional space. Particle filters did not give sufficiently accurate results so one needs to find a different approach.

In our particular problem, a discontinuity of second order, where the limits of the integrand function from the left and from the right to the discontinuity point do not have the same value, is present in the integrand, as can be seen in Figure 3.4 in the previous chapter. It is reasonable to assume that the problem could have a multitude

of discontinuities, so an approximation method which uses smooth functions may not be the best choice in this instance. We present in section 4.2, a deterministic method of integration which will be improved and applied to the continuous state Conditional Random Fields training in section 4.3. This, we hope, will allow us to correctly compute the value of the each integral in the sequence and assessing the local and total potentials. We will present our results in section 4.5 before we draw our conclusions in 4.6.

## 4.2 Binary Space Partition and Integration

The Riemann integration requires a form of partitioning. We will use a particular way of partitioning called Binary Space Partition (BSP) [Fuchs 80]. This will allow us to partition only relevant parts of the space and create a binary tree which implicitly generates a indexed partition.

The error of an integral approximation can be upper bounded by the *supremum* of the absolute value of derivative multiplied by half of the square of the interval. If a function is integrable in the Riemann sense then

$$\left| \int_a^b f(x)\,dx - (b-a)f(a) \right| \leq \frac{(b-a)^2}{2} \sup_{a \leq x \leq b} \left| f'(x) \right| \tag{4.2.1}$$

This can be deduced from the mean value theorem but a more general theorem can be found in [Priestley 97]. Since the Riemann integral is a linear operator, we can write a defined integral over a domain as sum of the integral on sub-domains covering the initial one.

$$\int_a^b f(x)\,dx = \sum_{i=0}^{k} \int_{a_i}^{a_{i+1}} f(x)dx \tag{4.2.2}$$

We can then apply another bounding formula to each one of the sub-integrals, where an upper bound is given by the summation of each of upper bounds of each sub-integral.

$$\left| \int_a^b f(x)dx \right| \leq |b-a| \sup_{a \leq x \leq b} |f(x)| \tag{4.2.3}$$

In the case where a point $\kappa$ is added to one of the intervals and the interval is split in two

then

$$(a_{i+1} - a_i) \sup_{a_i \leq x \leq a_{i+1}} f(x) \geq (\kappa - a_i) \sup_{a_i \leq x \leq \kappa} f(x) + (a_{i+1} - \kappa) \sup_{\kappa \leq x \leq a_{i+1}} f(x) \qquad (4.2.4)$$

either the split generates the same upper bound or it generates new, lower, upper bound because the *supremum* of the function in one of the sub-intervals is lower than the *supremum* in the initial interval. This is consequence of the definition of the *supremum*. In the first case no gain in precision has been obtained but in the second we have decreased the error of the approximation of the Riemann integration towards the correct value of the integral. By recurrence, the argument can be applied for a countable number of intervals. Lets remember that we can demonstrate the increase in precision of the integration by adding a point using this process. If one consider that both of the sub-interval have the same *supremum* as the initial interval and they are equal to $S$ then it is simple to see that $S(\kappa - a_i) + S(a_{i+1} - \kappa) = S(a_{i+1} - a_i)$. So by the definition of *supremum*, the *supremum* of either sub-interval is lower or equal than the *supremum* from the initial interval and in consequence the inequality is generated.

The main idea behind *Adaptive Quadrature* integration [Evans 00] is using some sort of information to split the interval in a point which give the best improvement. Adaptive quadrature also defines a recurrence on the method until a minimum value for a predetermined evaluation of the error of the integral have been achieved.

The difficulty is to choose the best place to split the interval in question. Unless there is some way, by a way of an algorithm, to find the correct point of interest, which implies some previous information about the integrand, we only can devise a strategy which will be sub-optimal.

One strategy is to generate an isometric partition where the size of each sub-interval is equal. This strategy has the virtue of being simple and is commonly used for didactic or demonstration purposes, but it is not hard to see that it does not take in to account the information obtained in each function evaluation to assess where the most relevant parts of the function lie or in other words where the most relevant splitting points are located.

Suppose that we have some minimal information about the integrating function, by using, for example, a small number of points generated using a isometric grid or partition,

where $a_i$ are the boundary points of the intervals:

$$y_i = f(x_i) \,:\, i \in N \qquad \text{where} \qquad x_{i+1} - x_i = const. \qquad \text{and} \qquad x_i = a_i + \frac{a_{i+1} - a_i}{2} \tag{4.2.5}$$

It is possible now to calculate a numerical approximation of the derivative.

$$\dot{y_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \approx \frac{df(x)}{dx} \tag{4.2.6}$$

and we assume that the calculated numerical derivative is representative of the *supremum* of the derivative in each interval

$$\dot{y_i} \approx \sup_{a_i \leq x \leq a_{i+1}} \left| f'(x) \right| \tag{4.2.7}$$

By locating the maximum of the *supremum* on every interval, a guess can be made about which interval the error for the integral has the highest bound,

$$\max_i \left[ \frac{(a_{i+1} - a_i)^2}{2} \, \dot{y_i} \right] \approx \max_j \left| \int_{a_{j+1}}^{a_j} f(x)dx - Q_j \right| \tag{4.2.8}$$

Even if no certainty can be assessed, this interval has the highest potential to generate the greatest change in the total error in the integral with the information we have obtained. So a recurrence of this concept can be made until a stopping criteria is reached.

A numerical implementation of this idea generates some small issues that should be addressed. Binary splitting of the interval, even if it appears as the most straight forward approach, may not be the most interesting strategy. Let us suppose that we have a set of $N$ intervals $\bigcup\{[a_i, a_{i+1}]\}$, and a set of evaluation points $\{x_i : x_i = a_i + \frac{a_{i+1} - a_i}{2}\}$ located in the middle of each interval. Since we have assessed the derivative by using two evaluation points located in two different adjacent intervals, either we change the boundary points for these two integrals to accommodate a new interval, or the intervals are split together. In the former case, the points we evaluate in the parent intervals will no longer be located in the middle of the intervals after splitting, which might produce errors since the evaluations of these integrals might be needed again and in the opposite direction. If, as in the latter case, the splitting is done using a Binary Space Partition (BSP) scheme,

where each interval $[a_i, a_{i+1}]$ is split into two intervals $[a_i, a_{i+1/2}]$ and $[a_{i+1/2}, a_{i+1}]$, the number of new evaluation points will be four; two for each two parent intervals. However the evaluation points for the parent intervals will be located in the same place as the new boundary point dividing the new intervals, so we have to abandon these points. This leads to a new scheme of splitting based on a Ternary Space Partition (TSP) as a generalisation of the BSP, where instead of splitting each interval in two, we split the interval in three. This new scheme will still require two new evaluations for each split interval, but will retain the one evaluation point from the parent interval. So a gain in precision is achieved with the need to evaluate of no more than two points, the same as BSP where one of the points had to be discarded. This gain is only obtained for a one-dimensional space, for a higher order integral the advantage will be lost, for example: for a two-dimension integration for each split in a generalised BSP, we need four more evaluation points and we discard one, but for a TSP we need 8 more evolution points for the gain of one.

In figure 4.1 we illustrate a TSP partition. The intervals were chosen by assessing the derivative represented by the red line between two evaluating points, represented by black discs, defined in the previous iteration, and with black lines representing the boundaries. For exemplification purposes we split the left interval using a TSP scheme and the right interval using a BSP one. The interval split using the BSP scheme, represented on the right in our figure, was split in two through a boundary point represented by a green dashed line. This generates two more interval in the middle of each two new evaluation points, represented by the two green circles, which need to be evaluated. But since the boundary line locations splitting the interval coincide with the location of the black point representing the evaluation point from the previous iteration, these points must be discarded since we only retain the points in the middle of the intervals. On the other hand, the left interval, corresponding to the TSP, was partitioned in three through two boundary points represented by the blue dashed line. This generates three evaluation points represented by the two blue circles and the the black circle filled in blue. The location of the black circle filled with blue, which needs to be evaluated, coincide with the location of the previous iteration evaluation point so its value can be retained leaving only the need to evaluate the two blue points. In conclusion with the TSP we have gain a more precise mesh without any increase in the evaluating cost.

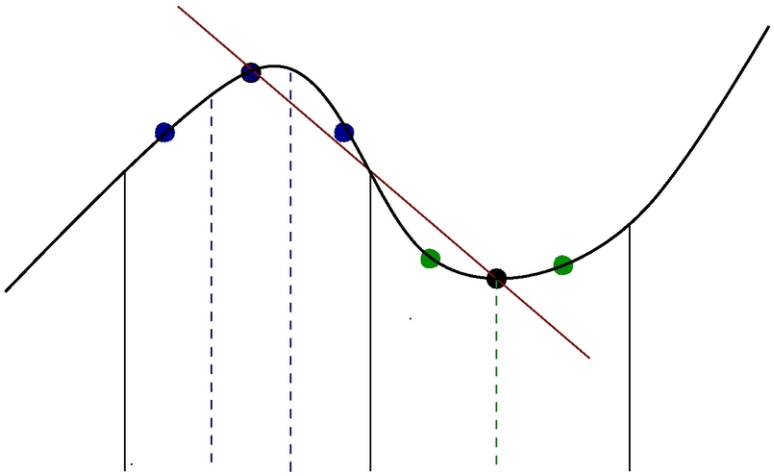

Figure 4.1: Example of a TSP partition.

This integration strategy has been summarised in Algorithm 1 for a one dimensional integration of a function $f(x)$. In lines the 1 to 7 the evaluations points and the intervals are initialised using an isometric distribution where the points are located at equal distance. The lowest boundary points for the first interval and the highest for the last coincide with the limits of integration. At line 8 the integral is calculated based on the initial partition of the space defined on the previous lines. At line 9 the value for the previous integral is defined in such a way that it will never meet the stopping criteria. Line 10 initialises a *while* cycle where the stopping criteria is defined to be if the relative error between the previous integral and the current integral falls below a threshold $r$, the cycle is finished. In this case the algorithm returns at line 22 the value of the calculated integral $I$, the intervals $p$, and the evaluation points $x$. If the stopping criteria is not met then the current integral is reassigned to the previous integral in line 11. The derivatives are evaluated by calculating the difference between the values of the functions at the consecutive evaluation points divided by the distance between the evaluation points on lines 12 to 14. At line 15 the index for the maximum of the values of the derivatives is found. At line 16 the new intervals calculated using the index obtained from the maximum derivative are appended to the set of the previous intervals. The weights affecting the difference between the boundary values of the intervals will place the new boundary values in the correct place following the TSP scheme. The same process is applied to both the new evaluation points in line 17 and the value of the function at this new evaluation points in line 18. At line 19 the number of evaluation points is updated. Finally at line 20 the new integral is computed

before a new cycle is initiated at line 21.

Let us define the integral of $f(x)$ in the interval $[a, b]$ using the TSP integration as $T_{sp}\int_a^b f(x)dx$. In fact the TSP integration, as it has been defined, is no more a linear operator with domain and co-domains $L \to \mathbb{R}$, assuming $L$ as the space of Riemann integrable functions. This is the consequence of the stopping criteria, since we need to remember that TSP is a adaptive process while Riemman integration is not. We will see an extension of TSP from a one-dimension function to a bi-dimension function in the next chapter.

To test TSP integration *versus* the Riemann integration with a isometric grid we have run a simple integration on the function

$$f(x) = \begin{cases} \exp(-\frac{x^2}{0.5^2}) & \text{if } x > -0.5 \\ 0 & \text{otherwise} \end{cases} \tag{4.2.9}$$

in the interval $[-2, 2]$. The results are presented in Table 4.1. This results can be compared with the value of 0.8165255223 obtained using the Maple$^{TM}$ package [Maple 10]. Even if both integration processes have good approximations, the TSP integration presents a better value than the one obtained with the Riemann integration, if compared with the result obtained from the symbolic manipulator software package.

| **# Points** | **TSP** | **Riemann** | **Maple$^{TM}$** |
|---|---|---|---|
| 53 | 0.810617811467 | 0.813345279536 | 0.8165255223 |
| 101 | 0.818221321733 | 0.814783394027 | 0.8165255223 |
| 153 | 0.817540005344 | 0.820103007543 | 0.8165255223 |
| 201 | 0.817135611110 | 0.819254037252 | 0.8165255223 |
| 245 | 0.817202004896 | 0.815788090263 | 0.8165255223 |

Table 4.1: Results for the TSP and Riemann integration test.

It is not hard to see that TSP integration will be more interesting compared to the Riemann integration in the case where the values of the function close to the singularity are higher compared to the remaining of the values of the function. The TSP will focus its attention on this region so instead of the boundary of the singularity be discriminated by only a standard interval as in the case of the Riemann integration the TSP integration

**Algorithm 1** Calculate $\int_a^b f(x)\,dx$

**Require:** $[a, b], n, r$

1: **for** $i = 1$ to $n + 1$ **do**
2: $\quad p_i \leftarrow a + i \cdot (b - a)/(n + 1)$
3: **end for**
4: **for** $i = 1$ to $n$ **do**
5: $\quad x_i \leftarrow a + (1/2 + i) \cdot (b - a)/(n + 1)$
6: $\quad y_i \leftarrow f(x_i)$
7: **end for**
8: $I \leftarrow \sum_i^n y_i \cdot (b - a)/n$
9: $I_{prev.} \leftarrow I(1 - r) - 1$
10: **while** $(I - I_{prev.})/I > r$ **do**
11: $\quad I_{prev.} \leftarrow I$
12: $\quad$ **for** $i = 1$ to $n$ **do**
13: $\quad\quad y'_i \leftarrow (y_{i+1} - y_i)/(x_{i+1} - x_i)$
14: $\quad$ **end for**
15: $\quad j \leftarrow \max_i y'_i$
16: $\quad p \leftarrow p + \{[a_j, a_j + 1/3(a_{j+1} - a_j)]\} + \{[a_j + 2/3(a_{j+1} - a_j), a_{j+1}]\} + \{[a_{j+1}, a_{j+1} + 1/3(a_{j+2} - a_{j+1})]\} + \{[a_{j+1} + 2/3(a_{j+2} - a_{j+1}), a_{j+2}]\}$
17: $\quad x \leftarrow x + \{a_j + 1/6(a_{j+1} - a_i)\} + \{a_j + 5/3(a_{j+1} - a_j)\} + \{a_{j+1} + 1/6(a_{j+2} - a_{j+1})\} + \{a_{j+1} + 5/6(a_{j+2} - a_{j+1})\}$
18: $\quad y \leftarrow y + \{f(a_j + 1/6(a_{j+1} - a_i))\} + \{f(a_j + 5/3(a_{j+1} - a_j))\} + \{f(a_{j+1} + 1/6(a_{j+2} - a_{j+1}))\} + \{f(a_{j+1} + 5/6(a_{j+2} - a_{j+1}))\}$
19: $\quad n \leftarrow n + 4$
20: $\quad I \leftarrow \sum_i^n y_i \cdot (a_{i+1} - a_i)$
21: **end while**
22: **return** $I, p, x$

will discriminate that region with interval until this interval will not significantly change the value of the integral.

One disadvantage of this integration process is that the convergence is not assured. To demonstrate this affirmation let us give a counter-example where in the initial step we have a situation where, even if the functions require further partition, the algorithm will halt.

Assuming an interval $[a, b]$ and an initial partition with $n$ points as $\cup_{i=0}^{n-1} [a + i(b - a)/(n - 1)]$ then the centre points will be located at $\mu_i = \{a + (b - a)/2(n - 1) + i(b - a)/(n - 1)\} : \forall i \in \{0, \ldots, n - 2\}$. Now if we consider a function defined by $f(x) = \sum_{i=0}^{n-1} c \cdot \exp\{-\mu_i/\sigma\}$ then the true value of the integral will be $\approx n$ but the algorithm will return $(b - a) \cdot c$. Since the maxima of the function, which have the same value, coincide with the evaluation points, the derivative calculated will be zero and the algorithm will not be able to partition further. In other words an initial decomposition of the space where the evaluating points coincide with the means of a mixture of Gaussians, represented by the function $f(x)$ since a mixture of Guassian is as sumation of Guassians [Bishop 07], will give a wrong evaluation of the integral of this mixture, and the error will increase with the standard deviation of the Gaussians. One possible solution to minimise the impact of this problem is to generate some noise around the evaluation points but this will burden the process in terms of calculating time. We will have to assume this as a limitation of the process and also assume that the functions are well behaved enough and that we have some previous knowledge to help us avoid this behavior. Every model has it own limitations, and the difficulty is to ensure the system comply with the constraints of the model. Usually it is possible to access that compliance by previous knowledge of the system. For example a trajectory of a projectile under Newtonian dynamics can be consider a continuous curve. It can be considered that other real systems have the same benign behavior. Adaptive quadrature is notoriously badly behaved in terms of classes of functions in which any adaptive methods is used. In the words of Evans & Swarts [Evans 00] “Adaptation is no guarantee of success either as integrands can always be constructed that defeat whatever adaptive strategy is devised”.

As it has been stated previously the splitting process either reduces the absolute value of the total error or does not produce any change. So we can define a sequence indexed

to the number of evaluating points to the absolute value of the total error, so since the sequence is positive and thus has a lower bound by the previous argument it will converge. The question arising is to which value the sequence will converge ? Or in other words if there is a class of functions which have a absolute total error sequence which will not converge to zero ? Other problems in the lack of assured convergence have its origins in how the stopping criterion is defined and in the way the implementation assess the derivatives. Even if these two issues look independent of each other they are closely related. The evaluation of the error is not correctly performed since we do not have the information of the derivative at all the points of the interval and in consequence to its *supremum*, so it is not possible to keep track of the total error, forcing us to rely on the comparison between two consecutive integrals evaluations. Also the rigidity with which the evaluation points are located might lead to a wrong evaluation of the error producing the results exemplified by our example, represented in Table 4.1.

It is for the reasons presented above that we can not assure a correct convergence without a formal proof of both issues; the convergence of the total absolute error sequence to zero and the correct evaluation of this error by the two consecutive integrals. But for this proof one needs to find the functions which behave properly under this limitations. Based on that knowledge and to close the issue under continuous state CRF there is the need to reformulate the conditions for the feature functions from this integrability conditions on the $\alpha$-$\beta$ recursions.

## 4.3 CRF with Ternary Space Partition Integration

The training of continuous observations-continuous state CRFs can now be implemented with the TSP Adaptive Integration scheme which has been presented previously. We now present how to obtain the governing equation for training our problem under this integration scheme. Using the equations derived in the previous chapter for the $\alpha^t(v)$ equation (3.3.16) and $\beta^t(s)$ equation (3.3.17) recursion to obtain the scaling factor $Z_\lambda(\mathbf{x})$ and the negative log-likelihood gradient $\nabla H$, where $H$ being the error function, let us re-write then under a TSP integration perspective. If, as we have defined before, the local potentials $\Psi$ are given by $\Psi^t(s,v) = \exp(\sum_{i=1}^{I} \lambda_i f_i(s,v,\mathbf{x},t))$ then each $\alpha$ is obtained from

the following recurrence formula:

$$\alpha^t(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s, v, \mathbf{x}, t)\right) \cdot \alpha^{t-1}(s)ds = \int \Psi^t(s, v) \cdot \alpha^{t-1}(s)ds \tag{4.3.1}$$

where the initial conditions are given by

$$\alpha^0(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(\bar{y}^0, v, \mathbf{x}, t)\right) d\bar{y}^0 \tag{4.3.2}$$

In the case that we have an initial distribution we can replace $\alpha^0(v)$ with this distribution. In the case that we don't have any information about the initial conditions we can assume a uniform distribution based on the principle of "Occam's razor", we integrate it by Riemann integration, for simplicity, using a simple isometric sample with $n_\alpha^0$ intervals

$$\alpha^0(v) = \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s^0, v, \mathbf{x}, 0)\right) ds^0 = \sum_{i=0}^{n^0} \Psi^0(s_i^0, v) \cdot \frac{(a-b)}{n_\alpha^0} \tag{4.3.3}$$

where $s_{i+1}^0 - s_i^0 = const.$ in all the domain $[a, b]$. Let's deduce the $\alpha^1(v)$

$$\alpha^1(v) = \int \left(\sum_{i=0}^{n_\alpha^0} \Psi^0(s_i^0, s^1) \cdot \frac{(a-b)}{n_\alpha^0}\right) \Psi^1(s_i^1, v)ds^1 \tag{4.3.4}$$

using a TSP integrating scheme

$$\begin{aligned}\alpha^1(v) &= T_{sp} \int \left(\sum_{j=0}^{n_\alpha^0} \Psi^0(s_j^0, s^1) \cdot \frac{(a-b)}{n_\alpha^0}\right) \Psi^1(s_i^1, v)ds^1 \\ &= \sum_{i=0}^{n_\alpha^1} \left(\sum_{i=0}^{n_\alpha^0} \Psi^0(s_j^0, s^1) \cdot \frac{(a-b)}{n_\alpha^0}\right) \Psi^1(s_j^1, v)(a_{i+1}^1 - a_i^1)\end{aligned} \tag{4.3.5}$$

So the discrete form of $\alpha^0$ will be

$$\alpha^0(s_i^1) = \sum_{i=0}^{n_\alpha^0} \Psi^0(s_j^0, s_i^1) \cdot \frac{(a-b)}{n_\alpha^0} \tag{4.3.6}$$

and $\alpha^1$

$$\alpha^1(s_i^2) = \sum_{j=0}^{n_\alpha^1} \Psi^1(s_j^1, s_i^2)(a_j^1 - a_{j+1}^1)\alpha^0(s_j^1) \tag{4.3.7}$$

so the generalization for any time step is given by

$$\alpha^t(s_i^{t+1}) = \sum_{j=0}^{n_\alpha^t} \Psi^t(s_j^t, s_i^{t+1})(a_j^t - a_{j+1}^t)\alpha^{t-1}(s_j^t) \tag{4.3.8}$$

Equally a beta expression can be obtained as

$$\beta^t(s_i) = \sum_{j=0}^{n^t} \Psi^t(s_j^t, s_i^{t+1})(b_j^{t+1} - b_{j+1}^{t+1})\beta^{t-1}(s_j^{t+1}) \tag{4.3.9}$$

Using the alpha beta expression is now possible to use $\nabla B$ from the expression (3.5.16) deduced in section 3.5

$$\frac{\partial B}{\partial \lambda_l} = \frac{1}{Z_\lambda(\mathbf{x})} \sum_{t=1}^{T} \int\int \alpha^{t-1} \cdot \Psi^{t'}(s^{t-1}, v^t) \cdot f_l(s^{t-1}, v^t, \mathbf{x}, t) \cdot \beta^t \, ds^{t-1} dv^t \tag{4.3.10}$$

to obtain its discrete form.

In equations (4.3.8) and (4.3.9) the formulas for the discrete formulation of $\alpha^t$ and $\beta^t$ have been deduced, so we can re-write the previous equation by replacing the alpha and beta in a continuous form by the discretised form to obtain a computable formulation for the gradient of $B$:

$$\frac{\partial B}{\partial \lambda_l} \approx \frac{1}{Z_\lambda(\mathbf{x})} \sum_{t=1}^{T} \sum_{i=1}^{n_\alpha^t} \sum_{j=1}^{n_\beta^t} \alpha^{t-1}(s_i^t) \cdot \Psi^{t'}(s_i^{t-1}, v_j^t) \cdot f_l(s_i^{t-1}, v_j^t, \mathbf{x}, t) \cdot \beta^t(v_j^{t+1}) \tag{4.3.11}$$

One important aspect of TSP integration is the sensibility of the resultant integration to the location of the evaluation points and the intervals. The wrong position of the evaluation points will lead to a poor performance of the summation on the integration process. This is important since with the TSP adaptive integration each integrand needs its own space partition. With Riemann integration the only relevant feature is the number of interval but with TSP integration the location and dimension of the intervals is also relevant. So it is important to evaluate the correct grid of evaluation points and intervals for each function, since this is the point of the integration scheme. The issue with this formulation is that we have obtained the correct grid integration scheme for the $\alpha$-$\beta$ recursion with the function $\Psi(s, v, \mathbf{y}, t)$, but we have not correctly obtained the grid for calculating the integral for the expectation of the feature functions given by $\Psi(s, v, \mathbf{y}, t) \cdot$

$f_i(s, v, \mathbf{y}, t)$. If the number of feature functions were just one it will need a single forward-backward pass, but the correct grid evaluation for the expectation of the feature functions will need to calculate at least the same number of passes as the number of feature functions, since each feature function will generate a new integrand.

But the correct calculation of equation (4.3.10) creates more difficulties than just an increase in the number of $\alpha$-$\beta$ recursions. From this equation we can see not only that we have an integration of $\Psi(s, v, \mathbf{y}, t) \cdot f_i(s, v, \mathbf{y}, t)$ but we also need to integrate it with $\alpha(s)$ and $\beta(v)$, due to the interconnected integration variables, so even if we have a correct grid for the expectation of the feature functions integral evaluations we need to re-evaluate the $\alpha(s)$ and $\beta(v)$ for that grid given the previous alpha-beta discretised grid. The $\alpha$ and $\beta$ where calculated using their specific grid and the specific grid for the expectation of a feature function will most probably not match that grid so to correctly calculate the total integral one needs a grid which gives a good integration with alpha beta and the expectation evaluation function.

So to correctly compute the integral, the generation of a sequence of correct intervals and evaluation points for the expectation of the feature functions function is needed. This can be done by using the TSP method to evaluate the integral of this function using the grid obtained from the previous time step evaluation as in the alpha recursion. This can be represented by

$$
\begin{aligned}
{}_{E}^{l}o_j^{t+1} = \underset{{}_{E}^{l}a_j^{t+1}}{\operatorname{argmin}} \Bigg| \sum_i \sum_j \Big[ \Psi({}_{E}^{l}s_i^t, {}_{E}s_j^{t+1}, \mathbf{y}, t) \cdot f_i({}_{E}^{l}s_i^t, {}_{E}^{l}s_j^{t+1}, \mathbf{y}, t)({}_{E}^{l}a_{i+1}^t - {}_{E}^{l}a_i^t) \\
\cdot({}_{E}^{l}a_{j+1}^{t+1} - {}_{E}^{l}a_j^{t+1}) \Big] - \int \Psi(s, v, \mathbf{y}, t) \cdot f_i(s, v, \mathbf{y}, t) ds\, dv \Bigg| \quad (4.3.12)
\end{aligned}
$$

where ${}_{E}^{l}a_i^t$ are the boundaries of each interval $i$ of the grid at time $t$ and for the feature function $l$, the evaluation points ${}_{E}^{l}s_i^t = ({}_{E}^{l}a_{i+1}^t - {}_{E}^{l}a_i^t)/2$ are the middle of each interval. This equation represents the required grid for the correct evaluation of the integral.

So for each feature function we obtain a grid at each time step which will correctly assess the expectation of the feature functions. Now we need to combine this new grid with the alpha recursion. This can be done by generating a new alpha which contains the

information from the previous time steps correctly integrated with only $\Psi(s, v, \mathbf{y}, t)$ like

$$ {}^{l}_{E}\alpha({}^{l}_{E}s_j^{t+1}) = \sum_i \sum_j \left[ \Psi(s_i^t, {}^{l}_{E}s_j^{t+1}, \mathbf{y}, t)(a_{i+1}^t - a_i^t) \right] \tag{4.3.13} $$

However the last integral will not be correctly computed because the last grid was optimized for a different function; the expectation of feature function $l$. Since, by increasing the number of points, the accuracy of the integral evaluation will not reduce, we can correct this problem by generating a new grid using both sets of points; one optimized for $\Psi(s, v, \mathbf{y}, t)$ and one optimized for $\Psi(s, v, \mathbf{y}, t) f_k(s, v, \mathbf{y}, t)$. So the new grid for the time step $t$ will be given by the union of the two grids

$$ {}^{l}_{E+}a^t = \{{}^{l}_{E}a^t\} \cup \{a^t\} \tag{4.3.14} $$

again we can re-write equation (4.3.13) as

$$ {}^{l}_{E+}\alpha({}^{l}_{E+}s_j^{t+1}) = \sum_i \sum_j \left[ \Psi(s_i^t, {}^{l}_{E+}s_j^{t+1}, \mathbf{y}, t)(a_{i+1}^t - a_i^t) \right] \tag{4.3.15} $$

and equation (4.3.10) as

$$ \begin{aligned} \frac{\partial B}{\partial \lambda_l} \approx \frac{1}{Z_\lambda(\mathbf{x})} \sum_{t=1}^{T} \sum_{i=1}^{n_\alpha^t} \sum_{j=1}^{n_\beta^t} {}^{l}_{E+}\alpha^{t-1}({}^{l}_{E+}s_i^t) \cdot \Psi^{t'}({}^{l}_{E+}s^{t-1}, {}^{l}_{E+}v^t) \\ \cdot f_l({}^{l}_{E+}s^{t-1}, {}^{l}_{E+}v^t, \mathbf{x}, t) \cdot \beta^t({}^{l}_{E+}v_j^{t+1}) \end{aligned} \tag{4.3.16} $$

Since the equations for a backward pass are similar to the ones we have deduced previously, we will not repeat the procedure here.

With this scheme to correctly compute the expectation of each feature function, we have increased the number of recursions needed from 1 to the number of feature functions + 1. Even if it is possible to calculate all the grids and evaluation points and the alpha and beta support for both grids in a single forward and backward pass the number of operations will remain the same as the grid calculation with several passes.

We have thus manage to deduce a set of formulae producing the gradient of the negative log-likelihood or error function via an adaptive integration scheme using a TSP method

to train continuous states continuous observations CRFs. Also we have seen the difficulty of correctly assess all the necessary integrals since this will lead to an increased number of operations.

## 4.4 Regularisation

In the real world, even if we have training data with both state and observation sequences for supervised learning, the data is usually obtained with noise on the measurements. The risk when training models with noise then, is the possibility to over-fit the data. During the process of training, the model does not yet discriminate between the proper behaviour of the system generating the data and the discrepancies resulting from the noise. This ignorance may lead to the assumption of the noise being part of the signal, and, in consequence, the model tries to fit not only the system but also the noise leading to over-fitting [Bishop 07].

To prevent this behaviour the usual solution is to add a penalty factor to the log-likelihood which forces the parameters to remain close to the origin. Maintaining the parameters close to the origin is equivalent to increasing the rigidity of the model preventing it to fit the noise. So any penalty factor in the form of $||\lambda - \mathbf{0}||$, or $||\lambda||$ for simplicity, with any norm should be acceptable, but the choice of the norm produces other consequences which we will talk about in Chapter 6, so the usual norm used is the square Euclidean norm $||\lambda||^2 = \sum_i \lambda_i^2$ [Bishop 07]. Several types of regularisation have been tested in Conditional Random fields [Smith 05]. Since the results showed no relevant difference, the Euclidean norm representing a Gaussian prior will be used.

A new degree of freedom must be added, in the form of a free factor $\eta$ to allow a homothetic transform of the regularization factor. To correctly choose this new factor we need to use two sets of training data; one with which we train the model with several values of $\eta$, and one in each we test its best value in a cross validation. The log-likelihood for the training data is expected to decrease as $\eta$ decreases, but in the validation log-likelihood we expect a maximum point after which the log-likelihood starts to decrease again as the variable decreases, this point is an estimate of the value for $\eta$.

So in our continuous state conditional random fields model the error function corre-

sponding to the negative log-likelihood will be

$$H = -\log\left[\frac{1}{Z_\lambda(\mathbf{y})}\int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(x^{t-1}, x^t, \mathbf{y}, t)\right) d\mathbf{x}\right] + \eta\sum_{i=1}^{I}\lambda_i^2 \tag{4.4.1}$$

and its derivative in with respect to $\lambda_i$, since

$$\frac{\partial}{\partial\lambda_i}\left[\eta\sum_i \lambda_i^2\right] = 2\eta\,\lambda_i \tag{4.4.2}$$

will be

$$\begin{aligned}\frac{\partial H}{\partial\lambda_i} = \frac{1}{Z_\lambda(\mathbf{y})}\int_{\mathbb{R}^T} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(s^{t-1}, s^t, \mathbf{y}, t)\right)\cdot\sum_{t'=1}^{T} f_i(s^{t'-1}, s^{t'}, \mathbf{y}, t)\,ds \\ + \sum_{t=1}^{T} f_i(x^{t-1}, x^t, \mathbf{y}, t) + 2\eta\,\lambda_i\end{aligned} \tag{4.4.3}$$

which represents the equations for continuous state conditional random fields with a regularization factor.

## 4.5 Results

In section 4.3 we have deduced the equations for the error function and its derivative. We have also shown that to correctly evaluate the value of the derivative, a substantial number of operations need to be performed; a number proportional to the number of feature functions plus one. The advantage we hope to gain by using a Newton-like minimisation method might be overcome by this very large number of operations needed for obtaining the derivatives. Knowing the fact that the expectation of the feature functions will be incorrectly evaluated we have nevertheless acknowledged the need to try the Newton-like method for a fast convergence. So we have decided to use the derivatives equation (4.3.16) with only the grid to correctly integrate $\Psi(s, v, \mathbf{y}, t)$ but not $\Psi(s, v, \mathbf{y}, t)f_k(s, v, \mathbf{y}, t)$ and test its convergence.

To test this new integration method we used the same problem defined in the previous chapter with synthetic data generated *a priori*, but this time with noise generated by a Gaussian with zero mean and standard deviation of 0.5. All the tests were performed with a set of initial condition of $\lambda = [10.0, 10.0, 10.0]$ and with a precision parameter $r = 0.1$

for the integration stopping criterion, with the feature functions given by Table 3.1 and the domain of integration being $[-2, 2]$.

We start by verifying the sampling of the Ternary Space Partition scheme by plotting a graph of a complete $\Psi(s, v, \mathbf{y}, t)$ transition.

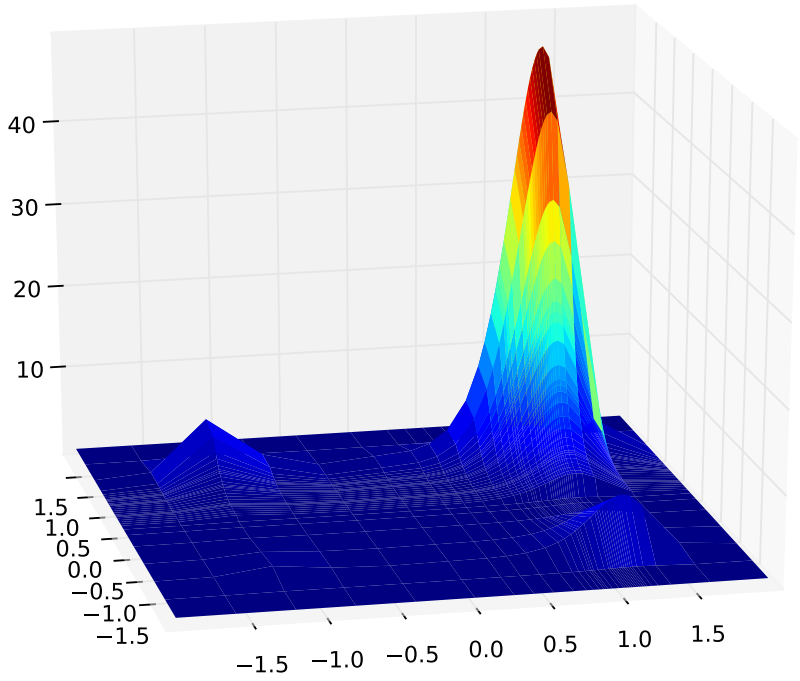


Figure 4.2: Example of a local transition potential.

A local potential, as it has been defined, has a domain in two dimensions and a co-domain in $\mathbb{R}$ for the one dimensional problem. We run an $\alpha$ recursion to acquire the sampling points for the two integrations; $t-1$ and $t$, and use this sampling to calculate the local $\Psi$ presented in Figure 4.2. As can be seen in Figure 4.2 the white lines corresponding to the sampling points of each time step have a greater density when they intersect the plane perpendicular to the lines where in the curve the values of the function and of the gradient are more relevant or larger. This indicates that our adaptive integration scheme is producing the expected results, since it is sampling more where the function is more relevant in terms of the TSP integration scheme, where the function is larger and has big derivatives.

Using this scheme to produce an error function $H$ and then using the minimizing

algorithm, we managed to obtain a convergence of the model parameters. We then used a Viterbi algorithm with a simple isometric distribution, to produce an approximation of the states for this same example, the results are presented in Figure 4.3.

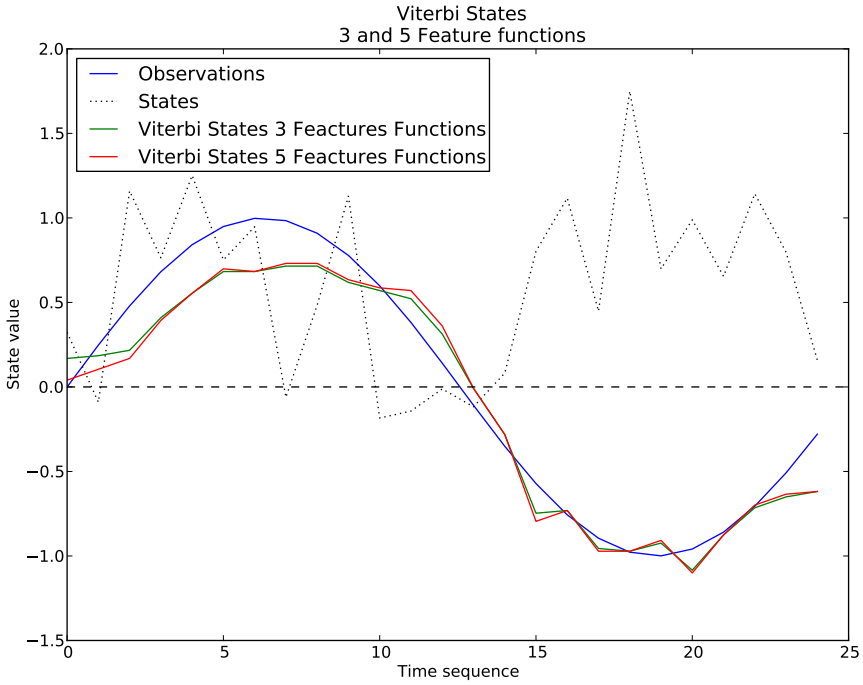


Figure 4.3: Observations and results on the state space.

As we can see, both sets $\{\#1, \#2\}$ of feature functions are in a good concordance with their resulting values. This indicates that firstly, the two extra feature functions are redundant, and secondly, since we obtained convergence for both sets, the results obtained in terms of parameters are reliable.

We also can see how this problem is non-trivial even thus it initially looked to the contrary. The values for the observations should be positive and thous located only on the upper plane, except for the noise. Nevertheless, with the extra information we supplied via the third feature function, the model was able to discriminate between the two signs, following the observations and filtering the signal.

Tests for the convergence for this problem with five types of minimisation methods were performed, four from the Scipy optimization package and one from BLMVM package. The four from Scipy package are two simple function minimises, fmin which minimises the function using the NelderMead downhill simplex algorithm, and fmin_powell

which uses a modified Powell's method, and two Newton-like methods with the gradient calculated with TSP and by it's own internal mechanism of finite differences for calculating the gradient; fmin_bfgs using the uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, and fmin_cg, an implementation of a non-linear conjugate gradient algorithm minimiser [Jones 01]. The last method we used was BLMVM which is a implementation of the limited memory variable metrics quasi-Newton optimisation from [Benson 01].

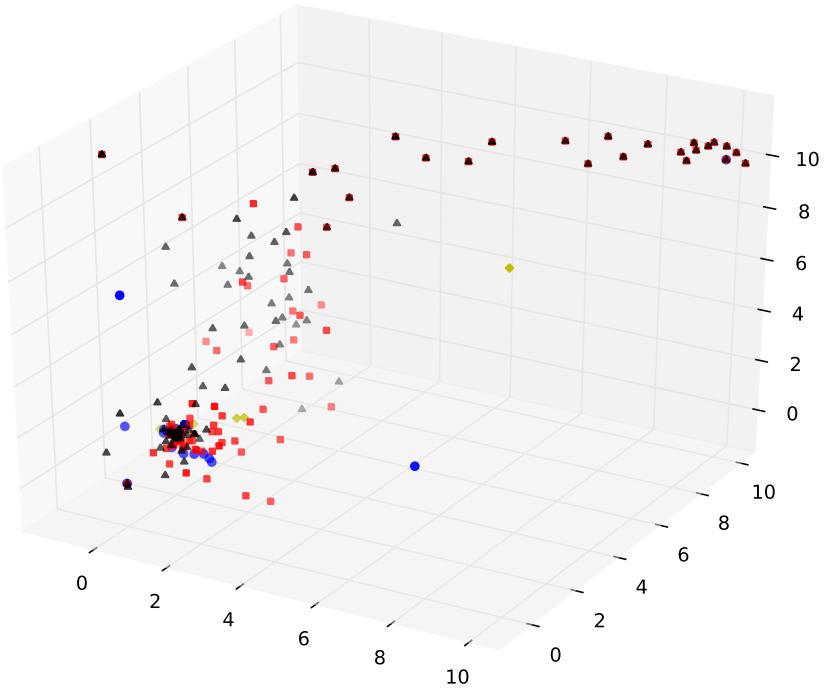


Figure 4.4: Trajectory of the parameters in the parameters space.

All the tests with the different minimizing algorithms converged to within the same neighbourhood, as it is presented in Figure 4.4. The black dots represent the evolution of the parameters in the parameter space from the TSP integration scheme and minimisation's with the fmin algorithm, the red dots for a Riemann type integration with 200 intervals with the fmin algorithm, the yellow dots for TSP with the LMVM algorithm for minimisation and finally the blue dots represents again a Riemann integration with 200 intervals and using the LMVM algorithm. The results obtained by the Viterbi algorithm were indistinguishable as can be seen in Figure 4.3, for both of them.

A search surrounding this neighbourhood of the parameters reveals that the graphics representing the results produced by the Viterbi algorithm are very similar to the ones

produced from the parameters inside the neighbourhood, indicating a shallow negative log-likelihood. Another consequence found when using TSP derivative scheme in the Newton-type minimizing algorithms is after a certain amount of time steps or sequence length the number of iterations started to increase and the minimizing algorithms started to produce warnings about the lack of precision.

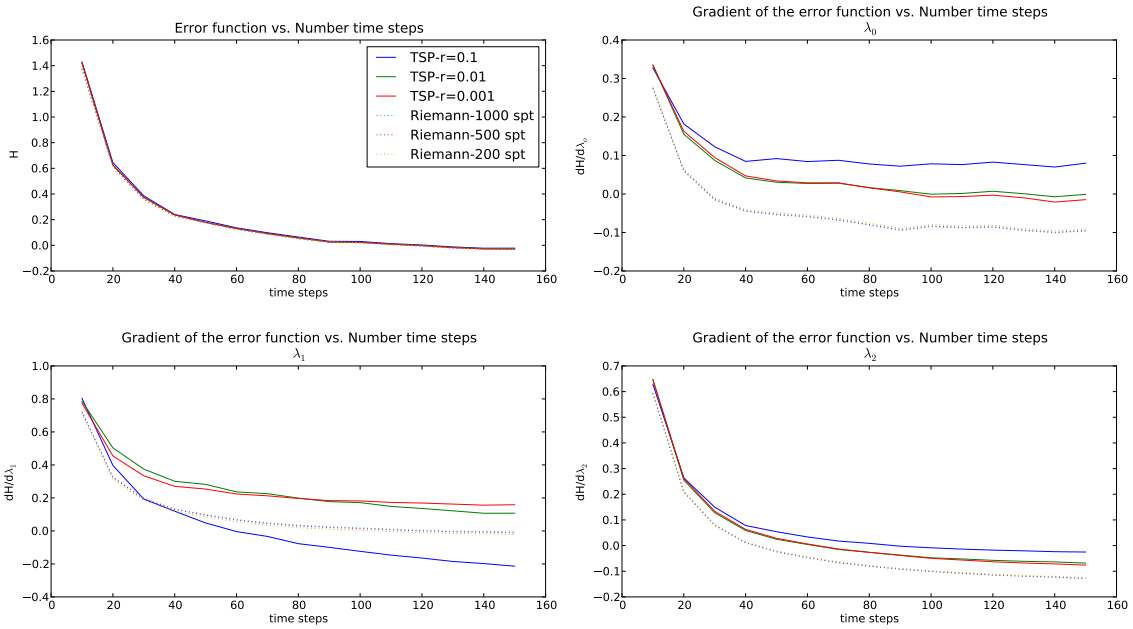


Figure 4.5: Negative log-likelihood and gradient for TSP and Riemann integration over a range of lengths of the training sequence.

The variation of the error functions and its gradient components have been computed for the same problem and for several time steps or sequence lengths between 10 and 150, with the TSP integration scheme and with a Riemann type of integration, for comparison purposes, with three increasing levels of discrimination $r = \{0.1, 0.01, 0.001\}$ for TSP and $200, 500$ and 1000 intervals for the Riemann integration. As it can be seen in Figure 4.5 there is an agreement between all the results for the error function, and to a lesser degree for the third feature function, but the feature functions for filtering and tracking do present an accentuated divergence. These results are not unexpected since we know there is an incorrect evaluation the expectation of the feature functions in consequence of the inadequacy of the grid to evaluate this function integral.

Another further indication of this inadequacy to evaluate the derivative was produced

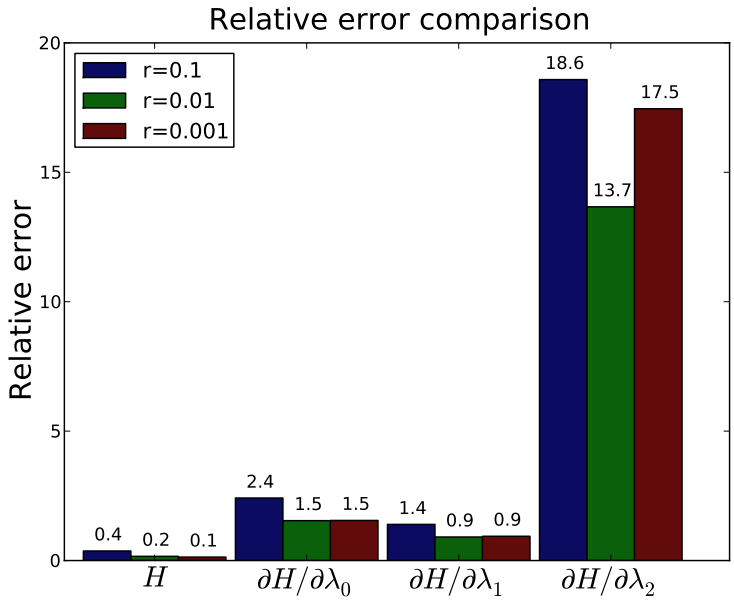


Figure 4.6: Additive inverse of the error function and its gradient.

by calculating the mean relative error between the sequences of values for the error function and it derivatives, in relation to the parameters obtained by the two methods, TSP and a Riemann integration. The relative error being the quotient of the absolute error and the values of the variable. It was assumed the Riemann integration with 1000 intervals as the standard to measure the relative error. We have compared the three sequences obtained with the three discrimination factors for $r$ with the defined standard. The results can be seen in Figure 4.6. The values for all the derivatives are at least half an order of magnitude above the values of the error functions, which is a strong indication of the further difficulty of calculating the derivative. The decrease in the error of the error function $H$ values as the values of the discrimination factor $r$ decrease, are a good indication that, in this case, the TSP integrations are producing the expected results. In the derivative calculation, on the other hand, as the discrimination factor decreases, the values of the derivative decrease in the case of feature function $f_0$ and $f_1$ but not in the case of feature function $f_2$. The values for the feature function $f_2$ in all the discrimination factors $r$ are an order of magnitude above the values for the other feature function derivatives. In our opinion these are strong indications to confirm the inadequacy of the calculation of the derivatives with the sign feature functions calculation as the least precise one. Both feature functions $f_0$ and $f_1$ return a continuous value so it is not unexpected they have a better behaviour

than feature function $f_2$ which represents the discontinuity of the potentials.

## 4.6 Conclusions

As it has been shown in the previous section, the idea of training continuous states conditional random fields with a TSP for adaptive integrations has not proved as successful as might be expected, even if TSP has successfully trained continuous states conditional random fields. All five of the optimisers manage to converge and to find a set of parameters all in a close enough neighbourhood. As the theoretical results show, to correctly calculate the feature functions expectation with a TSP scheme, one needs to perform an intractably large number of operations such that the use of a Newton-like minimizing method might not be of any advantage. The approximation made by using a small number of operations, corresponding to a grid optimized to integrate the local potential only, is successful until a certain point, since there is limit on the length of the training sequence the Newton-like minimizing algorithms are able to converge.

The theoretical deductions clearly state that if a correct integration is performed, then the error function and its derivative are consistent, and the only difference between discrete states conditional random fields and continuous states continuous conditional random fields are that the integrations replace the summations. Having errors produced by insufficient precision of the integration, will lead to poor evaluation of the factors and poor evaluation of the expectation of the feature functions and to the poor evaluations of the derivative.

To correctly train conditional random fields by maximizing the log-likelihood, one needs to calculate a sequence of local potential which will generate, by its product, the total potential. A small error in one of these factors is propagated through all of the sequence, and, if all the factors produce a small error then this error is additive (not in the strict sense since it follows a distributive formula [Chapra 06]). It is expected, then, that by having incorrect evaluations of the factors, the error at the end of a long enough sequence will be significant. The problem is exacerbated in the derivative calculations because, not only do we have a greater number of integrations and operations but also in the error function $H$, we have a logarithm function which is not present in the derivative

formula, which has the effect of damping the error as opposed to the exponential which has the effect of amplification. If we consider Kalman filters, their numerical stability comes from the fact that no exponentiation is involved, it relies only on multiplication and additions of its parameters. Conditional random fields impose some operations on the exponential and do not simplify as well as Gaussians, since the summation or the integration must be done on the exponentials before calculation of the logarithm as it can be seen in equation (3.3.3).

One other pertinent point of the results presented in the previous section is the choice of the feature functions. All of the feature functions we used in our example are functional, even if somewhat redundant as in the case of the #2 set. This produces a robustness of the most probable sequence of the parameters and it is the reason we obtained very similar results, in terms of the Viterbi algorithm resultant sequence, to the ones obtained in the neighbourhood of the convergence. So we need to conclude that a problem with conditional random fields is not only a question of correctly computing a set of parameters is also the need to correctly choose a good set of feature functions. This may lead to greater difficulty when dealing with a real and complex problem as opposed to a synthetic one, as in our test case.

Based on the previous presented results and arguments the important next steps would be to assure that a convergence is obtained for longer sequences through the Newton-like algorithms by using the full range of evaluation points obtained from the potential $\alpha$-$\beta$ pass and the product of the potential with each feature function. An interesting point will also be to compare the evaluating time and number of iterations *versus* the efficiency gain with both methods. This however will not be a definitive proof since it will be based on a particular case. Only a full intensive research on the constraints of the integration will give a definitive result.

# Chapter 5

# Bi-dimensional Continuous States Conditional Random Fields

## 5.1 Introduction

In the previous chapter we have presented the continuous state Conditional Random Fields in one-dimensional, in this chapter we will try to extended this concept to a two-dimensional case. The model presented will be virtually identical to the one-dimension one but with an increase complexity due to the increased dimension.

In two dimensions, the Ternary Space Partition concept we have presented will no longer be valid so we will present an extension of the Binary Space Partition concept instead, called *quadtrees*. This will retain the same interesting properties from the Ternary Space Partition of only increasing the precision of the partition where required.

The notation used in the previous chapters will be maintained but now each element of the sequence of states $\mathbf{x} = \{x^0, \ldots, x^t, \ldots, x^T\}$ will have dimensionality of two, $x^t \in \mathbb{R}^2$. In the case of the sequences of observations this limit is defined by the problem since it is only relevant for the feature functions and in consequence is independent of the model.

We will see in the next section the formal requirements on how to extend the continuous state conditional random fields formulation to a bi-dimensional space, we also briefly present in section 5.2 the quadtrees concept and in section 5.3 how to do adaptive integration with quadtree and in the latter two sections we will present our results and finally draw conclusions in section 5.5.

## 5.2 Quadtrees

A quadtree [de Berg 97] is a method for dividing a bi-dimensional space in an ordered way by using the property of isomorphism between the initial space partition and its component results, or in other words they have the same shape. This allows quadtrees to generate an ordered tree by recursion.

The concept of quatrees can be extended to higher dimensions. We shall use one bi-dimensional case as a example. By starting with a square we can split it in four quarters; splitting in two halves by a line orthogonal to two of the boundaries lines, and each of these halves again in half by a line orthogonal to the initial splitting line. We end up with four squares with 1/4 of the area of the initial squares. Each one of this sub-squares can be split again on is own without taking account any of the adjacent squares. This allows us to generate an ordered but asymmetric partition of the space. The union of all the subspaces generates the initial square, and each one of the sub-partitions is an isomorph of the initial one and therefore allowing a partition by recurrence. An example of this asymmetric partition can be seen in Figure 5.1.

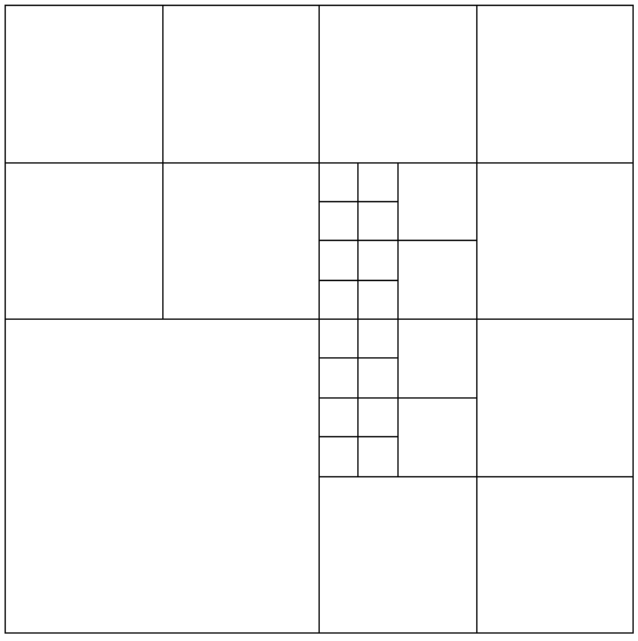

Figure 5.1: Example of a asymmetric partition of a square using quadtrees.

Let us note the difference between this partition and a partition generated by a number of equal-area squares. Let us take Figure 5.1 again, as a example, in this case the partition

based on an isometric partition will generate 256 small sub-squares instead of a partition of 47 sub-squares as represented in the Figure 5.1. If one does not need the refinement in certain places of the initial square space, this method can be advantageous since it will generate fewer partition points. We can see clearly that this type of partition can replace, in two dimensions, the Binary Space Partition we presented in the previous chapter, where the need of precision is located in the places of high values of the derivatives and it is where greater variations of the integral can be expected by the variation of the mesh refinement.

Quadtrees partitions schemes have been applied successfully to other problems which required dimensions higher than one and precision located in certain relevant regions [Kim 03]. A good example is its application to the finite elements method [Tabarraei 08], [Marchal 09], [Owen 98]. In the majority of the cases, finite elements methods problems are relevant on the study of the areas were the flux variable has is greater values, usually an iteration process of meshing and solving is required to refine the mesh in this areas. Due to the intrinsic process of the solving algorithm maintaining the previous mesh in the areas of poor interest is relevant to speed up the process. Quadtrees have the right properties and are well adapted to this requirements since it only refines the mesh at the relevant areas and keeps the remaining areas unchanged.

The quadtree concept is easily scalable since extensions can be found for higher dimensions like octrees for a three dimensions problem, *etc* ...

## 5.3 Adaptive Integration with Quadtrees

It has been presented in the previous section that quadtrees is well suited to an adaptive integration scheme [Min 07], since they extend the BSP partition scheme to a bi-dimensional space retaining the same properties as BSP. On the other hand performing an integration on a bounded bi-dimensional space, in the present case it will be restricted to a square for reason of simplicity. Since the key concept of adaptive integration is just to minimise the number of integration points by refining the mesh only were it is needed, quadtrees are well adapted to that process.

The process has been summarised in Algorithm 2.

One starts by initializing the variable of the previous integral iteration with a very

**Algorithm 2** Calculate $\int f(x)\,dx$ for $x \in \mathbb{R}^2$ with quadtrees

**Require:** $n, \epsilon$ and dimention of the square
  Initialize *mesh* with a small isometric mesh of order $n$ and $I \leftarrow$ very small number
  **for** *square* in *mesh* **do**
    f(*square*)$\leftarrow$ Calculate the function value in *centre point*(*square*) of *square*
    *area*(*square*) $\leftarrow$ Calculate the area of the *square*
    $I_{prev.} \leftarrow I_{prev.} + f(square) \cdot area\,(square)$
  **end for**
  $I_{prev.} \leftarrow I$
  **while** $(I - I_{prev.})/I > r$ **do**
    $I_{prev.} \leftarrow I$
    $I \leftarrow 0$
    **for** *square* in *mesh* **do**
      **for** *neighbour square* in *mesh* adjacent to *square* **do**
        $\partial$f$\leftarrow$ | f(*square*) - f(*neighbour square*) |
        $\partial$r$\leftarrow$ | *centre point*(*square*) - *centre point*(*neighbour square*)|
        *value*(*square*, *neigbour square*) $\leftarrow \partial f/\partial r$
      **end for**
    **end for**
    *square*, *neigbour square* $\leftarrow$ argmax *value*(*square*, *neigbour square*)
    *mesh*$\leftarrow$ *mesh* + split(*square*) + split(*neigbour square*)
    **for** *square* in *mesh* **do**
      $I \leftarrow I + f(square) \cdot area\,(square)$
    **end for**
  **end while**
  **return** $I$, *mesh*

low values and by generating a small isometric partition of squares on the integration domain. The current integral value is evaluated by adding the calculated integrating function values on the centre points multiplied by the square area for all the squares in this initial partition.

The new value for the integral is compared with the value of the integral calculated during the previous iteration. If the difference falls below a certain threshold, then the value of the current integral and the information with the location of the centre points and the area of the squares are returned.

Since, in the first iteration, we have initialised the variable of the previous iteration with a very low value, this will force the program to perform the next step, where the derivatives of the function for each square in the directions of the lines joining the centre point of the square of reference and the adjacent square, is evaluated by calculation the absolute value of the difference of the value of the function divided by the distance between the two centre points.

We retain the two adjacent squares which generated the highest derivative, and perform a division of each of the squares into four smaller ones as in the quadtree scheme. The new integral is then re-evaluated and compared with the previous iteration integral, if the comparison failed then a new iteration is initialize.

## 5.4 Dimension extension on the Adaptive Integration Concept in CRF

To obtain the formulation allowing us to generate the code for the bi-dimensional continuous state Conditional Random Fields one needs to verify all the equations are valid for higher dimension states. The equations for Maximum Entropy and Maximum likelihood presented in the previous chapters are still valid for higher dimensions since no condition or constraint has been put on the dimensionality, so the deduction obtained for $\alpha$ and $\beta$ in Chapters 3 and 4 are still valid. But now the elements of $x^t$ is a vector of dimension two (for a bi-dimension space problem). In consequence the integral in the equations for the $\alpha$ and the $\beta$ recursions (3.3.8) (3.3.15), as the one in the equation representing the error function (3.3.3), the total potential (3.3.11), and the error functions gradient (3.5.1)

and (3.5.16) represent now an integration on the domain $[a,b] \times [a,b]$, were $a$ is the lower bound and $b$ the upper bound of the integration domain. For simplicity purposes we will exemplify with only the alpha recursion,

$$\begin{aligned}\alpha^t(v) &= \int \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s,v,\mathbf{y},t)\right) \cdot \alpha^{t-1}(s)ds \\ &= \int_{[a,b]\times[a,b]} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s,v,\mathbf{y},t)\right) \cdot \alpha^{t-1}(s)ds \end{aligned} \tag{5.4.1}$$

With the quadtree integration scheme we produce an integral and a partition with areas which due to the commutative property of the integral summations does not require any type of ordering so if the number of partitions is $n$, the equations (5.4.1) become

$$\begin{aligned}\alpha^t(v) &= \int_{[a,b]\times[a,b]} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(s,v,\mathbf{y},t)\right) \cdot \alpha^{t-1}(s)ds \\ &\approx \sum_{i=0}^{n} \exp\left(\sum_{i=1}^{I} \lambda_i f_i(c_i,v,\mathbf{y},t)\right) \cdot \alpha^{t-1}(c_i) \cdot A_i \end{aligned} \tag{5.4.2}$$

where $A_i$ is the area corresponding to the square with centre point $c_i$. The same is applicable to the other equations for the beta recursion, error function, its gradient and total potential.

A simple extension of the Viterbi algorithm with an isometric partition where the evaluation points are located in an orthogonal grid can be obtained based on the same principle.

In fact this formulation not only allows the extension to the bi-dimensional space but also to any arbitrary higher dimension.

## 5.5 Results

For the two-dimensional problem we have extended the problem created in the one dimension case. We used a *cosine* functions for the second dimension states and, by doing so, generate a helix around the origin and developing through the time dimension. The functions giving the observations are equal to the one-dimensional case and in both dimensions are the square of the corresponding observations, as it can be seen in the next

equations (5.5.1).

$$\begin{aligned} x_0^t &= \sin(t/4.0) \\ x_1^t &= \cos(t/4.0) \\ y_i^t &= (x_i^t)^2 + w_i \qquad i \in \{0,1\} \end{aligned} \tag{5.5.1}$$

where $w_i \sim \mathcal{N}(0,\sigma^2)$ $i \in \{0,1\}$ is sample from a Gaussian with mean zero and standard deviation $\sigma$

The feature functions were also extended to accommodate this new dimension. We have created three new feature functions, mimicking on the ones from the one-dimensional problem; one to track the states based on observations, one to perform the filtering action and finally one to process the sign problem we discussed in Chapter 3. They have been summarised in Table 5.1.

| $f_i = f(s, v, \mathbf{y}, t)$ |
|---|
| $f_0 = -((v_1)^2 - y_1^t)^2$ |
| $f_1 = -(s_1 - v_1)^2$ |
| $f_2 = -((v_2)^2 - y_2^t)^2$ |
| $f_3 = -(s_2 - v_2)^2$ |
| $f_4 = \begin{cases} 1 & \text{if } \text{sign}(v_1) = \text{sign}(\sin(t/4.0)) \\ 0 & \text{otherwise} \end{cases}$ |
| $f_5 = \begin{cases} 1 & \text{if } \text{sign}(v_2) = \text{sign}(\cos(t/4.0)) \\ 0 & \text{otherwise} \end{cases}$ |

Table 5.1: Feature functions for the bi-dimension testing problem.

This bi-dimensional problem can be reduced to two one-dimensional problems since there is no cross-correlation between the two state sequences. Nevertheless the problem has been treated as bi-dimensional so we can assess more easily the consequences of the increase in dimensionality. It is not clear that all high-dimensional problems can be transformed in several of one-dimensional problems projected into two dimensions; in fact everything indicates the contrary. It is very probable that this kind of separable problems represents only an infinitesimal part of the possible problem set. Further analysis is required to see which class of problems will allow such decomposition. Also this decomposition might

be possible to obtain by a solid rotation around the axis obtained by the analysis of the eigenvalues of the variance of the data might be able to supply some information on how to perform this rotation, followed by the projection on the axis. Another possible approach is based on the variance eigenvalues producing a referential, where the projection might produce the best possible results. The advantage of using $n\times$ one-dimension problems instead of a single $n$-dimensional one can be appreciated if we consider the possibility of distributed or parallel computing.

A set of bi-dimensional data has been generated to test the bi-dimensional quadtree integration scheme, and a model was trained with both a gradientless algorithm and a Newton-like method until convergence, the same used on the previous chapter for the one-dimension problem, limited memory variable metric. The results obtained for the parameters were used to find a maximum probability sequence though a bi-dimensional Viterbi algorithm are presented in Figure 5.2, which we think is representative of both solutions.

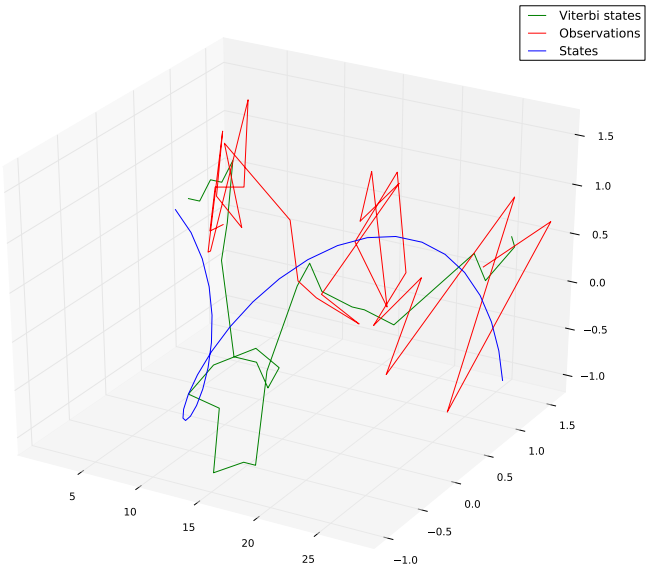


Figure 5.2: Observations and results in the bi-dimensional state space.

To help to compare this problem with the one-dimension problem we tested in the previous chapter, we have plotted the projection of the graph represented by a perspective projection in Figure 5.2 on the three axes as represented in Figure 5.3. All the initial conditions were set to $\lambda_i$ =1.1 with the precision parameter for the stopping criterion $r = 0.01$, the integration interval of [-2,2] for both dimensions, with 16 initial data points

for the first integration. The results obtained for parameters were $\lambda$ =[0.620, 1.244, 0.778, 1.224, 0.547, 0.660] obtained with the LMVM algorithm from the BLMVM package. The results obtained with the gradientless algorithm fmin from the Scipy package encoding the downhill simplex algorithm were similar. The first graphic in Figure 5.3 represents the first dimension of the state/observation sequence, equivalent to the problem in the previous chapter, the second graphic represents the state/observation sequence for the second dimension generated this time by a cosine, and third and last graphic represents the state/observation sequence for dimension one versus dimension two.

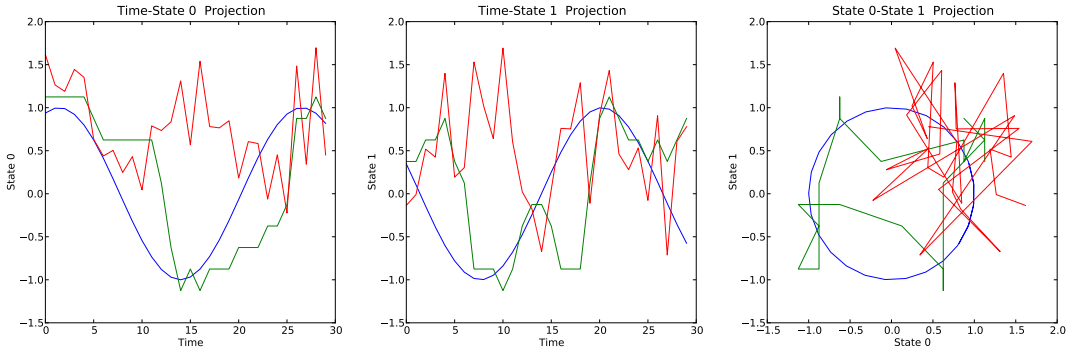


Figure 5.3: Projections of the observations and results in to the state-time planes and state-state plane.

As expected, the number of iterations needed to obtain convergence with a Newton-like minimizing algorithm in this bi-dimensional case was greater than for the one in only one dimension, and also the sequence length at wich we start to have convergence problems was shorter 25 time steps opposed to 75 in the one dimensional one, with stopping criterion equal to the ones used to train the one-dimension problem presented in the previous chapter.

The results in the first graphic from Figure 5.3 if compared with the results in Figure 4.3 show the discrepancy between the efficiency of both methods in recovering the states sequence given the observations. Assuming the two problems are the same, the one presented in the previous chapter and the projection of the current problem on the first dimension, which everything indicates they are, the increase in dimension, and though the different approach for training and inference including the increased difficulty in the two dimension discretisation compared to the one-dimensional one, indicates the integration process produces far worse results in the two dimensional case.

Nevertheless taking into consideration the noisy observations represented in red in Figures 5.3 and 5.2, we can consider that the model manages to track the states reasonably accurately and also assesses the sign correctly. Which means the bi-dimensional model manages to accommodate both the continuous process of tracking and filtering with the discrete process of sign assessing.

## 5.6 Conclusion

The extension of the one-dimensional problem to two-dimensions has generated the need to change the way the integrations were performed. We presented an extension of the TSP partition scheme we presented on the previous chapter which retains the same properties of low numbers of sampling points but is designed for a two-dimension problem. Quadtrees are scalable and thus can be extended to even more dimensions. This partition scheme allow us to maintain the same theoretical formulation obtained previously for a single dimension. We have also redesigned our one dimensional test problem to incorporate a second dimension.

The comparison of the results from this two-dimensional problem with the one with only one dimension showed the increased difficulty of solving the problem. This is not inherently by the problem itself since the problem proposed can be decomposed in two one-dimensional problems. This is because the methodology needed to incorporate the capability of solving truly bi-dimensional ones. We have to conclude that the extension in dimensionality of the integrations exacerbates the issues found in the previous chapter for the single dimension problem.

The aim of this chapter was to point out the possibility of extension of continuous state CRFs to higher dimensions with a bi-dimensional example with correct convergence. Even if a strong suspicion over the worse behaviour of the integration in the two-dimensional case must be assumed none of the presented results will support a definitive conclusion on the reasons of the discrepancies between the one-dimensional and bi-dimensional problem. For this a further investigation on the differences on the integration method over the different dimension is required, let us remember that in one-dimension it was used a TSP partition but in two dimensions it was used a extension of the BSP partitioning scheme.

So without any further information it is not possible to discriminate between the possible reasons, either the presence of convergence problems presented on the previous chapter for example, problems associated the change of partitioning scheme or other.

# Chapter 6

# Feature Functions Selection for Continuous State Conditional Random Fields

## 6.1 Introduction

The problem of variable and feature selection has generated an extensive body of work where several models have been developed to accommodate different approaches, models, and applications [Guyon 03]. The general problem can be seen as finding a method under the constraints of both applications and models to either increase the efficiency of the classification, or to reduce the number of operations required to perform this classification by reducing or transforming the space of features or variables. By reducing the number of features one can reduce the number of dimension of the problem. Also non relevant features can be a source of noise. We will try to select the most appropriate feature functions using only one feature functions selection method by the way of a regulariser.

A family of methods called *wrappers* generate a figure of merit for the definition of several sets of the variables which one may consider more relevant by performing a post-processing analysis of the classification results and an extensive search is required in all the possible subsets of features. The drawback of this approach is the high number of calculations needed to perform this task. Another difficulty may arise as to define this figure of merit.

Another family of methods called *filters* chose a subset of variables before any classification. This is performed by analysing the values of the variables and find properties or features which are discriminative or redundant between the variables.

*Embedded methods* are also a family of methods with perform the variable or feature analysis in conjunction with the training process.

In our case, we will not analyse the variables or the features but the feature functions which in our view is not strictly a feature selection even if it is a selection of a subset of variables the $\lambda$, nevertheless we will present a method which may be classified as an *embedded method*. Having some knowledge of the relevance of the past on the behavior of the system in the present is a important information. In systems with time delay, one relevant information is the knowledge of this delay, so by using a number of feature function covering a range of the delays and assessing the most relevant ones, we obtain important information from the data reflecting the system. This is what we tried to emulate in our testing problems.

Several methods for feature or subsets of features selection applied to Conditional Random Fileds can be found in the literature. McCallum [McCallum 03] proposes a induction process where a subset of feature functions, initialised with the empty set, are incremented with the feature function which presents the highest gain, while maintaining the other parameters as constant. The gain is the increase in the likelihood function. This method is compared with a gradient based method in [Chen 09]. The gain in the case of the gradient based method is obtained by using an approximation though the partial derivative of the likelihood. Also a comparison of methods is presented by Klinger [Klinger 09]. The comparison is made between three filter methods ranking the feature functions and an interactive feature pruning where a percentage of the lowest ranked features based on the $\lambda$ coefficients are removed from the subset until total depletion. Two publications [Day 06], [Dang 09] present genetic algorithms used as wrappers to perform feature function selection. The gene is composed of binary bits encoding the presence or absence of the feature function which index match the index of the bit. Vail and Veloso [Vail 08] presented a feature function selection algorithm using the $||\mathbf{x}||_1$ norm which promotes sparseness in the parameter vector $\lambda$.

We will present the general method in 6.2, its formulation in the continuous states

conditional random fields methodology in 6.3, the results in 6.4 and finally draw the conclusions in section 6.5.

## 6.2 Parameter shrinkage

We have said in previous sections that, outside the scope of predefined problems where we have some knowledge of the properties of the data generated in an artificial way, it might be difficult to define a correct set of feature functions. The capability of assessing the relevant functions, called parameter shrinkage, will allow us to extend the feature functions set to try several possibilities. One could use a *wrapper* method by performing a training process and an evaluation of the final results, using a norm on the difference of the states sequence for the training data and on the Viterbi generated sequence. But as we said previously this is computationally expensive. We will use some properties of the norms of the regularization factor instead.

If we consider a set of norms called p-norms as $||\mathbf{x}||_p = (\sum |x_i|^p)^{1/p}$, the only value of $p$ in which the equation $||\mathbf{x}|| = r$, were $r \in \mathbb{R}$, produces an isotropic result in a Euclidean space is the $||\mathbf{x}||_2$ norm or Euclidean norm, other norms produce a deformation of the Euclidean circle. The p-norms with the $p$ coefficient greater than two will produce a deformation where the Euclidean distance is higher at the two diagonal lines at $45^o$ in relation to the axes. If the order of the norm is lower than two then this distance will be shorter. In the particular cases were the norm is $||\mathbf{x}||_\infty$ the result is a square with edges parallel to the axes, and when $p = 1$ then it is a square where the vertices will be located on the axes. The order $p$ does not need to be a natural number it is also valid for any $\mathbb{R}_{1+}$.

We have presented in section 4.4 a regularization factor which we define to be the Euclidean norm. Minimizing the negative log-likelihood with a regularization factor can be seen as minimizing the negative log-likelihood function with a constraint on the parameters using the Lagrange multipliers [Bishop 07]. This constraint is equivalent to the norm of the parameter vector being forced to be less or equal to a constant in the minimisation process. The $\eta$ factor, the factor allowing the parametrisation of the regulariser or the weight affecting the regularizing function, is then a defined parameter which regulates the

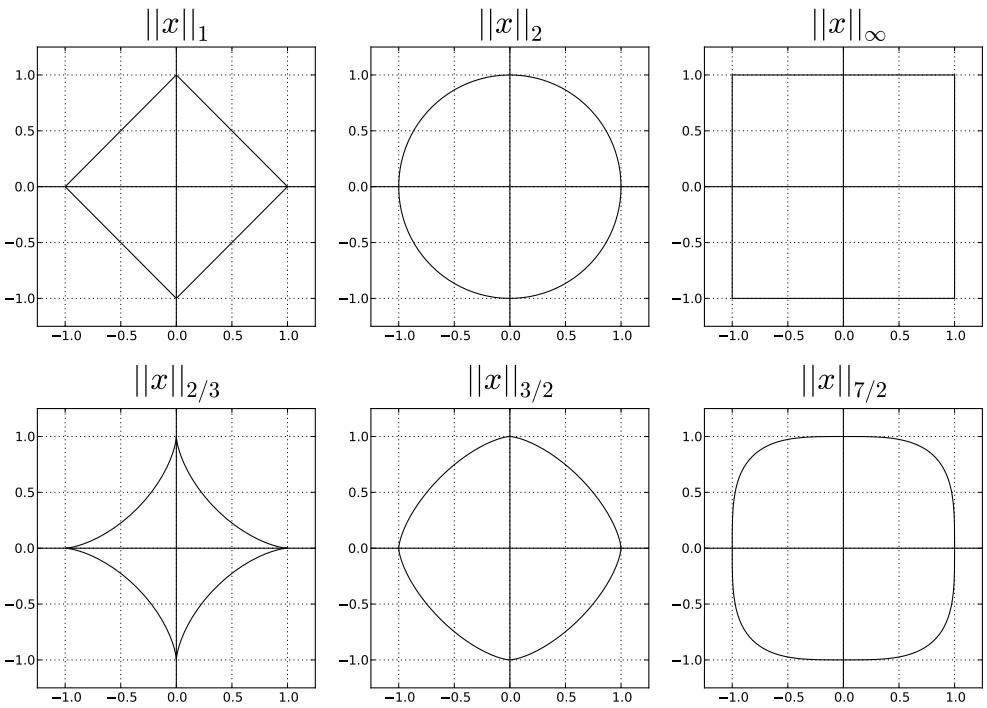


Figure 6.1: Examples of $||x||_p = 1$ for several values of p.

size of the vector norm. So the regularization is *per se* a parameter shrinkage process since it promotes the minimisation of the parameters. Even if all norms have this property they are not equal in their result, since some norms have the property of promoting even more uneven results in the parameters. We will used the term "parameter shrinkage" for the use of this norm, and not for the regularization process.

As the Euclidean norm is isotropic, all the values of the factors are treated in the same way. Now if we used a norm with $p$ lower than two, the regularization factor will penalise the vector of parameters farther away from the axis, by doing so it is promoting the vector to be closer to the axis. In consequence this forces the vector of parameters to be sparse, since if a vector lies close to the axis, some of it components are close to zero. Another way to see this effect is by considering equipotential lines around the minimum. The lower value equipotential lines will tend to intercept the unitary p-norm curve at its vertices or which lie close to the axes [Bishop 07].

Also note that the process of assessing the correct factor $\eta$ by calculating the log-likelihood of the model for a range of these parameters, but using a different set of training data, can be seen as a *wrapper* type of processing; but this time we are not trying to find

the relevance of the parameter, but instead we are trying to find the correct value for it.

Since all the norms for $1 < p < 2$ are expected to produce parameter shrinkage and there is an infinity of possibilities in the choice of norms, we have decided to use the $||\mathbf{x}||_1$ norm. The p-norm $||\mathbf{x}||_1$, also known as the "Manhattan norm" or "Taxicab norm", generates a regularisation coefficient known in the field as the *lasso* [Tibshirani 94]. This norm was successfully applied in discrete state conditional random fields for parameter shrinkage by [Vail 08] so we think it is a good candidate to produce correct results in the continuous state model.

One important point we need to stress is when we define the *function* $||\mathbf{x}||_p$ with $0 < p < 1$. This functions does not represent a norm any more, since it does not comply with the triangle inequality. In a strictly sense the notation $||\mathbf{x}||_p$ should not be applied as in Figure 6.1 for $p = 2/3$, but for purpose of clarity we have maintain this notation for $0 < p < 1$. An interesting point is we do not need specifically a norm so using this functions might be acceptable. But for their use one needed to verify the conformity of the consistency with the remain of the theory.

## 6.3 Lasso Regulariser in Continuous State Conditional Random Fields

The application of the $||\mathbf{x}||_1$ norm to continuous states conditional random fields is achieved by replacing the previous Euclidean norm in equations (4.4.1) by the new *lasso* regularisation.

$$H = -\log\left[\frac{1}{Z_\lambda(\mathbf{y})}\int \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{I}\lambda_i f_i(\bar{x}^{t-1}, \bar{x}^t, \mathbf{y}, t)\right) d\bar{\mathbf{x}}\right] + \eta\sum_{i=1}^{I}|\lambda_i| \qquad (6.3.1)$$

As we have seen in the previous two chapters, we will not use a gradient algorithm, but the application of the $||\mathbf{x}||_1$ norm to the gradient is also fairly simple to obtain for a small issue for the continuity of the derivative around zero [Guitton 03]. All that is required is

to replace the regularization factor in equation (6.3.1) by the equivalent Huber function:

$$M_\epsilon(\lambda_i) = \begin{cases} \dfrac{\lambda^2}{2\epsilon} & \text{if } 0 < |\lambda_i| < \epsilon \\ |\lambda_i| - \dfrac{\epsilon}{2} & \text{if } \epsilon < |\lambda_i| \end{cases}$$

This change of the regularization factor is needed to force the regularization factor derivative to be continuous. The derivative of $|x|$ is not defined at $x = 0$ since the limit of the left side derivative of $|x|$ is not the same as the limit of the right side

$$\lim_{x \to 0^-} \frac{d}{dx}|x| = -1 \neq \lim_{x \to 0^+} \frac{d}{dx}|x| = 1 \tag{6.3.2}$$

A necessary (but not sufficient) condition of a maximum or a minimum not in a boundary for a smooth function is defined by $\nabla H = 0$ so a continuous derivative is required. To prevent irregularities in the regularising factor and, as a consequence, the log-likelihood functional, which might cause problems in the numerical optimization, one forces the derivative to be continuous at zero and a very small and localised region around zero. This is done by changing the $||\mathbf{x}||_1$ norm in to the $||\mathbf{x}||_2$ when the value is less than $\epsilon$, since the derivative of $||\mathbf{x}||_2$ is continuous at zero. A important point is that the second derivative will not be continuous at the points $-\epsilon$ and $\epsilon$ but since this set has a Lebesgue measure inferior to the Lebesgue measure of the function, this may not be relevant to a minimisation method using the Hessian.

This small problem is only pertinent if a gradient based method is used. If a simple minimisation method which does not require the calculation of the derivative is planned, then the lasso may be used without any more considerations, which is our case.

## 6.4 Results

A problem suited for testing the feature functions selection in continuous CRF will, in preference, be a problem with a redundant feature function but with one feature function being more pertinent than the other. Using the problem set up in Chapter 4, using the downhill simplex algorithm from the Scipy package, with a extra feature function out of synchronisation with the current time will allow us to set up a few different problems based on the degree of synchronisation we used on the second feature function.

So the problem was defined by

$$
\begin{aligned}
x^t &= A(t) \cdot \sin(w(t) \cdot t) \\
y_0^t &= (x^t)^2 + w^t \\
y_1^t &= \begin{cases} 1 & \text{if } \sin(t/4.0) > 0 \\ -1 & \text{if } \sin(t/4.0) < 0 \end{cases}
\end{aligned} \tag{6.4.1}
$$

with the noise $w^t$ sample from a Gaussian with mean zero and standard deviation $\sigma = 0.5$, $w^t \sim \mathcal{N}(0, \sigma^2)$.

We have generated two problems based on equations (6.4.1) where the functions $A(t)$ and $w(t)$ are given in Table 6.1. The first problem is the same as the problem generated in Chapter 4 but this time the information about the signal is passed in a new dimension of the observations. In this problem the amplitude and frequency of the *sine* signal is constant as the previous problem. In the second problem we have increased the level of difficulty by varying the amplitude and the frequency. The range of this variation is between the amplitude and frequency from the first problem to its double. This change occurs in 50 time steps.

| **First problem** | **Second Problem** |
|---|---|
| $A(t) = 1$ | $A(t) = 0.02\,t + 1$ |
| $w(t) = 0.25$ | $w(t) = 0.05\,t + 0.25$ |

Table 6.1: Problems for the parameters shrinkage testing.

The feature function assessing the sign of the signal was modified to accommodate a shift in time, and the set was increased with a new feature function by duplicating this feature function for redundancy purposes as can be seen in Table 6.2. We have set up three different sets of feature function by assuming three different types of pairs of parameters controlling the amount of time shift summarised in Table 6.3

So for each of the two problems we will test three sets of parameters for the feature functions and we will compare, in all six cases, the lasso regularization factor which we hope will outperform the quadratic Euclidean norm used in the previous chapters. In

$$
\begin{array}{l}
f_i = f(s, v, \mathbf{y}, t) \\
\hline
f_0 = -((v)^2 - y_0^t)^2 \\
f_1 = -(s - v)^2 \\
f_2 = \begin{cases} 1 & \text{if } \operatorname{sign}(v) = y_1^{t-\tau} \\ 0 & \text{otherwise} \end{cases} \\
f_3 = \begin{cases} 1 & \text{if } \operatorname{sign}(v) = y_1^{t-\kappa} \\ 0 & \text{otherwise} \end{cases}
\end{array}
$$

Table 6.2: Feature functions for the parameters shrinkage testing.

| **First Set** | **Second Set** | **Third set** |
|---|---|---|
| $\tau = 0$ | $\tau = 0$ | $\tau = 1$ |
| $\kappa = 7$ | $\kappa = 1$ | $\kappa = 2$ |

Table 6.3: Parameters for the feature functions time shift.

all three problems presented, a redundant feature function will be created, but different information is supplied to the model, and supposedly one of them will supply more relevant or precise information. The information fed into the model through this feature function is the sign of the observations, since observations are sign-less, so if one of the feature functions supplies a greater number of correct sign states, it will be considered the most correct and the former will be considered the redundant one. In the third set of parameters the time parameter $\tau$ and $\kappa$ are so close together that the difference between the frequency of each of them in noisy observations can be easily equal or swapped. So it will not be totally unexpected if the $\lambda_3$ will be closer to zero than in the fourth one.

The results obtained are represented in Figure 6.2 for the first problem and Figure 6.3 for the second problem where we have plotted the value of $-\log \lambda_i$ for both norms. We can see represented tree different sub-figures, one for each set of parameters. In each sub-figures there is four sets of pairs of bars, each pair represent the value of the negative logarithm of the parameter for each feature function. The two different color represent the two different regularization functions. This transformation function, $-\log \lambda_i$, for changing the scale puts in evidence the values in the neighborhood of zero which we want to compare between the two norms since the objective is to obtain a value for $\lambda$ close to the zero for the redundant feature function.

In all the experiments both of the two norms perform well by correctly assessing the

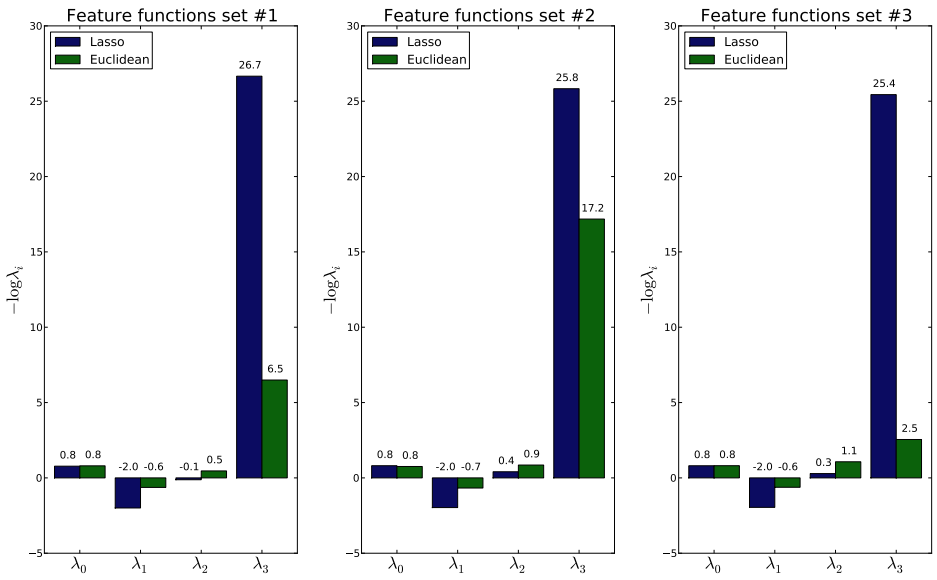


Figure 6.2: Value of the function $-\log\lambda$ for each feature function for problem #1

redundant feature function $f_3$. The lasso norm however always produces a $\lambda$ closer to zero except for the second set of the second problem. This is not unexpected since we said the Euclidean norm also tends to produce smaller parameters. In the case of the feature functions all being functional and independent, the Euclidean norm will be expected to perform well by correctly assessing the redundant function in the simpler cases. In the third set however the lasso regulariser shows better results than the Euclidean as expected since the problem requires a much finer precision and where the difference in shape of the iso-potentials should make a difference.

## 6.5 Conclusion

In the previous sections we have described how to obtain information to discard feature function which have less relevance to the model. This process is inherent to the method chosen to prevent the model from over fitting the data, since this method constrains the norm of the parameters vector to remain below a given threshold. A particular sub-set of the set of these regularizers produced an even more accentuated parameter shrinkage due to the shape of their isopotential lines. A particular element of this sub-set known as a lasso has be tested on discrete states conditional random fields with success [Vail 08], and we tried to assess its success in the continuous states case of this model.

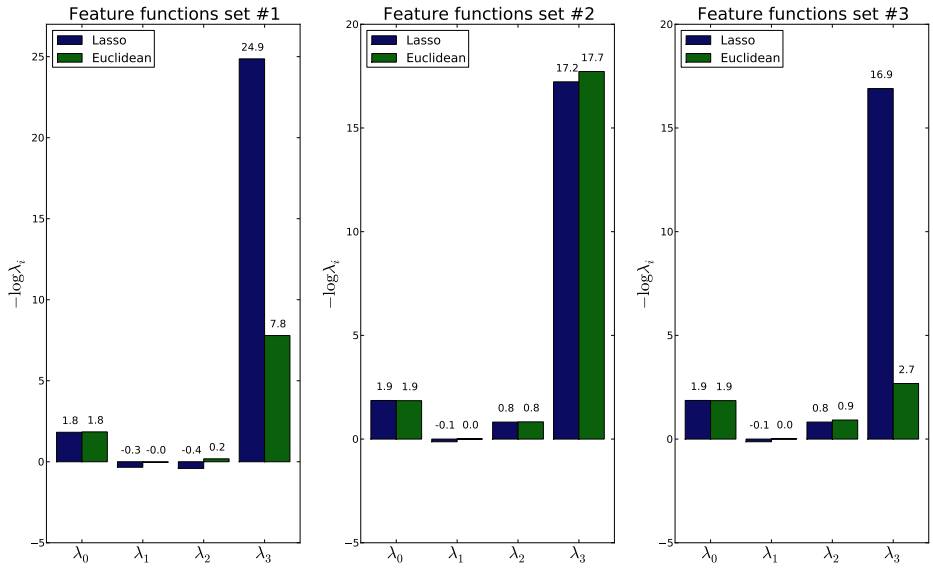


Figure 6.3: Value of the function $-\log\lambda$ for each feature function for problem #2

Based on the results obtained in our restricted problem we can say that the parameters shrinkage we obtained with the two regularisers but the lasso was more efficient in the most difficult case, since the results for the two problems were very similar, which indicates that feature functions selection are invariant to this type of changes.

# Chapter 7

# Summary, Conclusions and Future Work

## 7.1 Summary

In the previous chapters we have tried to extend discrete state conditional random fields into continuous states. For this we needed to deduce the equations regulating this extension based on solving the maximum entropy equation for continuous distributions, deducing the log-likelihood equations and it gradient, and thus formalising it through algebraic equations. This was possible by using the distributive commutative law of multiplication and the linearity of integration. Using these properties it is possible to aggregate the information from past or future sequences from a given index point in a function. This allowed us to generate an $\alpha$-$\beta$ recursion similar to other models, by integrating the local potential to the previous or posterior function, supporting either the forward or backward sequence. Differentiating this formulation in respect to the parameters, and using the same arguments we obtained the formalisation of the equation for the gradient.

The particle filter models were successfully used to train non-linear sequence models, so were good candidates to also train continuous states CRF. To assess this suitability one had to find the equation governing continuous conditional random fields under the particle filter perspective.

The first main idea behind this transformation is to not constrain the feature functions in a way where the HMM, Kalman Filter or some CRF were presented as an internal model

based on the previous observation and intended current observation value to produce a prediction and then another model using the intended current state value with the observations to perform the update producing the final model.

This led us to formulate a constraint on the predictive model by imposing its formulation outside the CRF model. The option chosen was to use a Gaussian distribution to perform this task since it is equivalent to a diffusion. By using a diffusion model we were assuming some implicit constraints, where the strongest could be considered to be the bound of the second derivative. The conditional random field equation can then be computed by using particle filter integrations on the potentials integrals. Particle filter integrations, as other stochastic integrations methods, have a very slow rate of convergence and we found out they are not the best combination with Conditional Random Field models since CRF are expensive in computing operations, and are not robust in handling errors. The errors from integration are reflected in the repeatability of the error function and its gradient, and this problem could not be overcome by the minimisation algorithm.

In an attempt to solve the integration issues uncovered while testing the training of continuous states conditional random fields with particle filters, we decided to try a deterministic type of integration but which incorporated the property of minimizing the number of evaluation points to reduce the associated computational costs. This was done by using adaptive integration, based on a sectioning of the integrating interval only in the locations where the maximum gain in precision is expected. The evaluation of the expected gain is obtained by analysing the numerical derivative of the integrand function. The new evaluating points location is performed around the points of greatest derivative. The adaptive scheme allows us to evaluate the integrals needed to train continuous states conditional random fields with a greater precision than the one obtained with particle filter integration and under a deterministic framework. Nevertheless the gain in precision was not enough to overcome all the problems. The Newton-like optimization algorithms were not robust enough to accept the errors of the derivative for longer sequences.

An extension to a bi-dimensional problem was easily obtained since the model equations remained the same. Nevertheless the algorithm assessing the location of the evaluating points needed to be augmented to accommodate the extra dimension this was obtained

by using the quadtrees concept. In the particular case of quadtrees they allow an increase in dimension to two but they are scalable using its extension to higher dimensions like octree in the tri-dimensional space, etc... We were then able to reproduce the training method as in the one-dimensional case. The results obtained were in some way similar to the previous case, since we were able to reach convergence with a Newton-like algorithm but with problems when the sequence reached a certain dimension. Comparing both cases the bi-dimension behaves in a less satisfactory way than the one dimension case.

Since the problems in Conditional Random Fields in general, and continuous states conditional random fields in particular, are not only dependent on the definition of states and observation but also on the definition of the feature functions, a correct definition of the model implies a correct definition of the feature functions. In simple cases this can be achieved with a minimum margin of error but for more complicated cases this definition may not be so simple. Having a method which could help the choice of the feature functions is consequently relevant. We used for this purpose a regularizer known to produce parameter vectors close to the axis. This helps to indicate the irrelevant feature functions since they are the ones with parameters close to zero. Since this kind of feature function selection was successfully used in Conditional Random Fields, we tested it in the continuous states model. The results indicate that this regularizer performs as expected and out-performs the most commonly used one in the most difficult case.

## 7.2 Conclusions

Continuous Random Fields are prone to numerical errors. The sequences of summations of exponentials required to compute the gradient of the additive inverse of the log-likelihood requires good precision in the calculations. In control theory it is a well known fact that derivation tends to amplify noise since, in that case, derivatives promote a high pass filter in the signal and in real applications the noise tends to be of higher frequency than the signal. But it is not totally unexpected that also the log-likelihood of the CRF model may suffer from the same phenomenon.

The difference between discrete state CRF and continuous state CRF lies in the integration process; grid filtering is in fact equivalent to Riemann integration to a discrete

model. Since discrete state CRF have been used successfully in several applications, we are forced to conclude that the integration process has to be responsible for the difficulties for the continuous states CRF training. This becomes apparent when the increase in the difficulty of the integration is augmented, as in the case where the results from the two-dimension problem were degraded in relation to the one dimension as the integration process got more difficult. Every thing in our results indicates the difficulty of training. When it is possible to obtain a convergence, continuous state condition random fields are exacerbated when using Newton-like methods.

Due to the inadequacy of the integration methods consequence of the computational cost to correctly integrate the expectation of each function at each time step in the two integrations tested we did not manage to solve the problem with a fast Newton like algorithm. Nevertheless we managed to train, with good results, continuous states conditional random fields in a single dimension, and to assess the correct feature function using a selective regulariser. The results obtained for the bi-dimensional case, even if a training was achieved, were not comparable with the single-dimension case. So we need to conclude continuous states conditional random fields can be trained without recourse to grid filters and the feature functions can be optimised, but their extension to higher dimensions is not evident.

## 7.3 Future Work

Our attempts to train continuous states conditional random fields raised some interesting questions related to either CRF or to adaptive integration of the sequence classification framework.

Since there is virtually no limitation on the feature functions for CRF either for discrete or continuous states, the model can, in theory, be quite extended. The first question raised is: is it possible to generalize CRF to continuous time ? Even if at first glance the need for this extension might not be apparent, the formulation of this generalized model will possibly allow us to better understand the connection between CRF and the approximation-interpolation theory.

Another important question about CRF is if there is a set, or sets, of predefined feature

functions with interesting properties which are able to replace a class of feature functions without these properties. We have seen the difficulty of assessing the expectation of the feature functions, if it is possible to find a series of feature functions which can approximate any linear combination of feature functions but which allow an analytical integration of the expectation formula.

We have briefly made a reference to the relation between the Viterbi algorithm and the Bellman principle but we have not mentioned the Pontryagin's maximum principle [Lewis 95] dual of the Bellman principle. It will be interesting to see CRF used under this framework, if is it indeed possible. Also Pontryagin's maximum principle allows continuous time models in an inherent way.

The electronic filtering process has been developed quite some time ago and there are some tools to help us compare and understand filters, even due to the flexibility of continuous states CRF obtained by the feature functions it would be interesting to analyse CRF under these methods.

Even if the TSP and quadtree integration schemes were not as successful as expected when training continuous states conditional random fields, we think they might be more successful in a less precision-demanding control application but with a demanding computer cost either from the function assessment or from higher dimensions.

As presented the regulariser associated with the p-norms has to have defined the value for p. But if we assume the p value as a parameter we might be able to find the optimal p for a given model or at least a better understanding of the behaviour of each norm and it impact as regularisers.

# References


[Baum 70] Leonard E. Baum, Ted Petrie, George Soules & Norman Weiss. *A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains.* Annals of Mathematical Statistics, vol. 41-1, pages 164–171, 1970.

[Benson 01] Steven J. Benson & Jorge J. More. *A Limited Memory Variable Metric Method in Subspaces and Bound Constrained Optimization Problems.* Technical report ANL/MCS-P909-0901, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[Bishop 07] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). Springer, 1 edition, October 2007.

[Borman 04] Sean Borman. *The Expectation Maximization Algorithm: A short tutorial.* Technical report, University of Notre Dame, 2004.

[Callen 85] Herbert B. Callen. Thermodynamics and an introduction to thermostatistics (2nd ed. ed.). John Wiley & Sons, Inc., 1985.

[Chapra 06] Steven C. Chapra & Raymond Canale. Numerical methods for engineers. McGraw-Hill Higher Education, 5 edition, 2006.

[Chen 09] Minmin Chen, Yixin Chen, Michael R. Brent & Aaron E. Tenney. *Gradient-Based Feature Selection for Conditional Random Fields and its Applications in Computational Genetics.* In Proceedings

of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence, ICTAI '09, pages 750–757. IEEE Computer Society, 2009.

[Dang 09] Thanh Hai Dang, Kristof Engelen, Pieter Meysman, Kathleen Marchal, Alain Verschoren & Kris Laukens. *Conditional Random Fields Feature Subset Selection Based on Genetic Algorithms for Phosphorylation Site Prediction.* Knowledge and Systems Engineering, International Conference on, vol. 0, pages 7–12, 2009.

[Day 06] Min-Yuh Day, Chun-Hung Lu, Chorng-Shyong Ong, Shih-Hung Wu & Wen-Lian Hsu. *Integrating genetic algorithms with conditional random fields to enhance question informer prediction.* In Proceedings of the 2006 IEEE International Conference on Information Reuse and Integration, IRI - 2006: Heuristic Systems Engineering, September 16-18, 2006, Waikoloa, Hawaii, USA, pages 414–419. IEEE Systems, Man, and Cybernetics Society, 2006.

[de Berg 97] Mark de Berg, Marc van Kreveld, Mark Overmars & Otfried Schwarzkopf. Computational geometry: Algorithms and applications. Springer, 1 edition, 1997.

[Dempster 77] A. P. Dempster, N. M. Laird & D. B. Rubin. *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society. Series B (Methodological), vol. 39, no. 1, pages 1–38, 1977.

[Dietterich 02] Thomas G. Dietterich. *Machine Learning for Sequential Data: A Review.* In Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, pages 15–30, London, UK, 2002. Springer-Verlag.

[Doucet 01] Arnaud Doucet, Nando de Freitas & Neil Gordon. Sequential monte carlo methods in practice (statistics for engineering and information science). Springer, June 2001.

[Doucet 08] Arnaud Doucet & Adam Johansen. *A Tutorial on Particle Filtering and Smoothing: Fifteen years Later.* Technical report, Department of Statistics, University of British Columbia, 2008.

[Duda 01] Richard O. Duda, Peter E. Hart & David G. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, 2 edition, November 2001.

[Evans 00] Michael Evans & Tim Swartz. Approximating integrals via monte carlo and deterministic methods. Oxford University Press, 1 edition, 2000.

[Friedland 96] Bernard Friedland. Advanced control system design. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[Fuchs 80] Henry Fuchs, Zvi M. Kedem & Bruce F. Naylor. *On visible surface generation by a priori tree structures.* In Computer Graphics, pages 124–133, 1980.

[Gregory 04] Michelle L. Gregory & Yasemin Altun. *Using conditional random fields to predict pitch accents in conversational speech.* In ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, page 677, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[Guedon 03] Yann Guedon. *Estimating Hidden Semi-Markov Chains from Discrete Sequences.* Journal of Computational and Graphical Statistics, vol. 12, no. 3, pages 604–639, 2003.

[Guitton 03] Antoine Guitton & William W. Symes. *Robust inversion of seismic data using the Huber norm.* Geophysics, vol. 68(4), pages 1310–1319, 2003.

[Guyon 03] Isabelle Guyon & Andre Elisseeff. *An introduction to variable and feature selection.* Journal of Machine Learning Research, vol. 3, pages 1157–1182, 2003.

[Hald 90] Aders Hald. A history of probability and statistics and their application before 1750. John Wiley & Sons, Inc., 1990.

[Hassan 05] Md. Rafiul Hassan & Baikunth Nath. *StockMarket Forecasting Using Hidden Markov Model: A New Approach.* In Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, ISDA '05, pages 192–196, Washington, DC, USA, 2005. IEEE Computer Society.

[Jaynes 03] Edwin T. Jaynes. Probability theory : the logic of science. Cambridge University Press, Cambridge, UK, 2003.

[Jones 01] Eric Jones, Travis Oliphant, Pearu Peterson*et al. SciPy: Open source scientific tools for Python*, 2001.

[Kalman 60] Rudolph Emil Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Transactions of the ASME–Journal of Basic Engineering, vol. 82, no. Series D, pages 35–45, 1960.

[Kim 03] Eui Joong Kim, Jacobo Bielak & Omar Ghattas. *Large-Scale Northridge Earthquake Simulation Using.* In 16th ASCE Engineering Mechanics Conference, pages 221–234. Springer-Verlag, 2003.

[Klinger 07] Roman Klinger & Katrin Tomanek. *Classical Probabilistic Models and Conditional Random Fields.* Technical report TR07-2-013, Technical Universitat Dortmund, Faculty of Computer Science, Algorithm Engineering (Ls11), 44221 Dortmund, Germany, December 2007.

[Klinger 09] Roman Klinger & Christoph M. Friedrich. *Feature Subset Selection in Conditional Random Fields for Named Entity Recognition.* In Proceedings of the International Conference RANLP-2009, pages 185–191, Borovets, Bulgaria, September 2009. Association for Computational Linguistics.

[Lafferty 01] John Lafferty, Andrew McCallum & Fernando Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* In International Conference on Machine Learning (ICML), pages –, 2001.

[Lewis 95] Frank L. Lewis & Vassilis L. Syrmos. Optimal control - 2nd edition. John Wiley & Sons, Inc., 1995.

[Limketkai 07] B. Limketkai, D. Fox & Lin Liao. *CRF-Filters: Discriminative Particle Filters for Sequential State Estimation.* In Robotics and Automation, 2007 IEEE International Conference on, pages 3142–3147, 2007.

[Maple 10] Waterloo Maple. *Maple 14 software package*, 2010.

[Marchal 09] Loc Marchal. *Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features.* In Brett W. Clark, editeur, Proceedings of the 18th International Meshing Roundtable, pages 65–84. Springer Berlin Heidelberg, 2009.

[McCallum 03] Andrew McCallum. *Efficiently Inducing Features of Conditional Random Fields.* In Christopher Meek & Uffe Kjærulff, editeurs, UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, August 7-10 2003, Acapulco, Mexico, pages 403–410. Morgan Kaufmann, 2003.

[Min 07] Chohong Min & Frédéric Gibou. *Geometric integration over irregular domains with application to level-set methods.* J. Comput. Phys., vol. 226, pages 1432–1443, October 2007.

[Murphy 01] Kevin P. Murphy. *An introduction to graphical models.* Technical report, Department of Computer Science, University of British Columbia, 2001.

[Ogata 87] Katsuhiko Ogata. Discrete-time control systems. Prentice-Hall, Inc., 1987.

[Oliver 04] Nuria Oliver, Ashutosh Garg & Eric Horvitz. *Layered representations for learning and inferring office activity from multiple sensory channels.* Comput. Vis. Image Underst., vol. 96, no. 2, pages 163–180, 2004.

[Owen 98] Steven J. Owen. *A Survey of Unstructured Mesh Generation Technology.* In INTERNATIONAL MESHING ROUNDTABLE, pages 239–267, 1998.

[Priestley 97] Hilary A. Priestley. Introduction to integration. Oxford University Press, 1 edition, 1997.

[Qi 05] Yuan Qi, Martin Szummer & Tom Minka. *Bayesian Conditional Random Fields.* In Robert G. Cowell & Zoubin Ghahramani, editeurs, Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, pages 269–276. Society for Artificial Intelligence and Statistics, 2005.

[Quattoni 07] Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins & Trevor Darrell. *Hidden Conditional Random Fields.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pages 1848–1852, 2007.

[Rabiner 89] Lawrence Rabiner. *A tutorial on HMM and selected applications in speech recognition.* Proceedings of the IEEE, vol. 77, no. 2, pages 257–286, February 1989.

[Sarawagi 04] Sunita Sarawagi & William W. Cohen. *Semi-Markov conditional random fields for information extraction.* In In Advances in Neural Information Processing Systems 17, pages 1185–1192, 2004.

[Schutter 99] Joris De Schutter, Jan De Geeter, Tine Lefebvre & Herman Bruyninckx. *Kalman Filters: A Tutorial.* Journal A-Benelux Quarterly Journal on Automatic Control, vol. 40(4), pages 538–546, 1999.

[Sha 03] Fei Sha & Fernando Pereira. *Shallow Parsing with Conditional Random Fields.* In Proceedings of HLT-NAACL 2003, pages 213–220. Association for Computational Linguistics, 2003.

[Shannon 48] Claude Shannon. *A mathematical theory of communication.* Bell System Technical Journal, vol. 27, pages 379–423 and 623–656, July and October 1948.

[Shen 07] Dou Shen, Jian tao Sun, Hua Li, Qiang Yang & Zheng Chen. *Document Summarization using Conditional Random Fields.* In Proceedings of IJCAI 07, 2007.

[Shigley 86] Joseph Edward Shigley. Mechanical engineering design. MacGraw-Hill Book Company, 1986.

[Smith 05] Andrew Smith & Miles Osborne. *Regularisation Techniques for Conditional Random Fields: Parameterised Versus Parameter-Free.* In Robert Dale, Kam-Fai Wong, Jian Su & Oi Yee Kwong, editeurs, Natural Language Processing IJCNLP 2005, volume 3651, pages 896–907. Springer Berlin / Heidelberg, 2005.

[Tabarraei 08] A. Tabarraei & N. Sukumar. *Extended finite element method on polygonal and quadtree meshes.* Computer Methods in Applied Mechanics and Engineering, vol. 197, no. 5, pages 425–438, 2008.

[Tappen 07] M.F. Tappen, C. Liu, E.H. Adelson & W.T. Freeman. *Learning Gaussian Conditional Random Fields for Low-Level Vision.* In CVPR07, pages 1–8, 2007.

[Taycher 06] L. Taycher, D. Demirdjian, T. Darrell & G. Shakhnarovich. *Conditional Random People: Tracking Humans with CRFs and Grid Filters.* In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 222–229, 2006.

[Tibshirani 94] Robert Tibshirani. *Regression Shrinkage and Selection Via the Lasso.* Journal of the Royal Statistical Society, Series B, vol. 58, pages 267–288, 1994.

[Vail 08] Douglas L. Vail & Manuela M. Veloso. *Feature selection for activity recognition in multi-robot domains.* In AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence, pages 1415–1420. AAAI Press, 2008.

[Vallespi-Gonzalez 08] Carlos Vallespi-Gonzalez & Tony Stentz. *Prior Data and Kernel Conditional Random Fields for Obstacle Detection.* In Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland, June 2008.

[Viterbi 67] A. Viterbi. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* Information Theory, IEEE Transactions on, vol. 13, no. 2, pages 260–269, January 1967.

[Wallach 04] Hanna M. Wallach. *Conditional random fields: An introduction.* Technical report MS-CIS-04-21, University of Pennsylvania, 2004.

[Welling 06] Max Welling & Sridevi Parise. *Bayesian Random Fields: The Bethe-Laplace Approximation.* In UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence. AUAI Press, 2006.

[Zienkiewicz 05] O. C. Zienkiewicz & R. L. Taylor. The finite element method for solid and structural mechanics (sixth edition). Elsevier Butterworth-Heinemann, 2005.

[Zweig 09] Geoffrey Zweig & Patrick Nguyen. *A Segmental CRF Approach to Large Vocabulary Continuous Speech Recognition.* Technical report, Microsoft Corporation, 2009.