Optimisation for POMDP-based Spoken Dialogue Systems

M. Gašić, F. Jurčíček, B. Thomson and S. Young
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, UK
{mg436, fj228, brmt2, sjy}@eng.cam.ac.uk

August 18, 2011

1 Introduction

Spoken dialogue systems (SDS) allow users to interact with a wide variety of information systems using speech as the primary, and often the only, communication medium. The principal elements of an SDS are a speech understanding component which converts each spoken input into an abstract semantic representation called a user dialogue act, a dialogue manager which responds to the user's input and generates a system act a_t in response, and a message generator which converts each system act back into speech. At each turn t, the system updates its state s_t and based on a policy π, it determines the next system act $a_t = \pi(s_t)$. The state consists of the variables needed to track the progress of the dialogue and the attribute values (often called slots) that determine the user's requirements. In conventional systems, the policy is usually defined by a flow chart with nodes representing states and actions, and arcs representing user inputs.

Despite steady progress over the last few decades in speech recognition technology, the process of converting conversational speech into words still incurs word error rates in the range 15% to 30% in realistic operating environments. Systems which interpret and respond to spoken commands must therefore implement dialogue strategies which account for the unreliability of the input and provide error checking and recovery mechanisms. As a consequence, conventional deterministic flowchart-based systems are typically expensive to build and fragile in operation.

During the last few years, a new approach to dialogue management has emerged based on the mathematical framework of partially observable Markov decision processes (POMDPs) [1, 2, 3]. This approach assumes that dialogue evolves as a Markov process i.e. starting in some initial state s_0, each subsequent state is modelled by a transition probability: $p(s_t|s_{t-1}, a_{t-1})$. The state s_t is not directly observable reflecting the uncertainty in the interpretation of user utterances; instead, at each turn, the system can observe only the noisy interpretation of the user input o_t with probability $p(o_t|s_t)$. The transition and observation probability functions are represented by a suitable stochastic model, called here the *dialogue model* $\mathcal{M}$.

The decision as to which action to take at each turn is determined by a second stochastic model, called here the *policy model* $\mathcal{P}$. As the dialogue progresses, a reward is assigned at each step designed to reflect the desired characteristics of the dialogue system. The overall goal is to maximise the expected accumulated sum of these rewards by optimising the dialogue model $\mathcal{M}$ and policy model $\mathcal{P}$, either on-line through interaction with users or off-line from a corpus of dialogues collected within the same domain. The representation of these models can be either parametric or non-parametric.

The ultimate performance of a statistical spoken dialogue therefore depends on how effectively the two models $\mathcal{M}$ and $\mathcal{P}$ can be optimised and the purpose of this chapter is to present recent advances in this area developed within the Dialogue Systems Group at Cambridge. For background, section 2 outlines in a little more detail the basic framework of POMDP-based dialogue systems and section 3 explains how a tractable stochastic dialogue model can be parameterised and implemented using dynamic Bayesian Networks[4]. The remaining sections focus on optimisation techniques. Section 4 explains how a more general form of inference called Expectation Propagation(EP), which can be viewed as a form of expectation-maximisation, can be used for both belief tracking and parameter optimisation [5, 6]. Section 5 explains how natural actor-critic reinforcement learning can be used to optimise the policy parameters $\mathcal{P}$ [7, 8], and how with a simple extension, it can also be used to optimise the dialogue model parameters $\mathcal{M}$ [9]. Finally, section 6 addresses the problem of fast on-line policy optimisation using Gaussian processes as a non-parametric policy model [10, 11, 12]. Section 7 wraps up with some general conclusions and pointers to future work.

2 POMDP-based Dialogue Management

The first key idea of POMDP-based dialogue management is that instead of computing the most likely state at each turn, the system tracks the probability of all states. The posterior distribution over states is called the belief state b_t and it is updated each turn using the current input observation o_t from the user. This update is referred to as *belief monitoring* or *belief tracking* and it is computed as follows

$$b_t = p(s_t|h_t;\mathcal{M}) = kp(o_t|s_t;\mathcal{M}) \sum_{s_{t-1}} p(s_t|s_{t-1}, a_{t-1};\mathcal{M}) b_{t-1} \tag{1}$$

where $h_t = \{a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t\}$ is the dialogue history, $\mathcal{M}$ is the chosen dialogue model and k is a normalising constant. Whilst simple in theory, the practical implementation of the Markov model underlying this belief monitoring process is complex. The belief state b_t is continuous and has very high dimension which makes direct implementation impossible. One solution is to group equivalent states into partitions and maintain only the N-best most likely partitions as is done in the HIS system [13]. While such a representation enables a more detailed expression of the dialogue state [14], it requires a non-parametric modelling of the transition and observation probabilities [15]. Another approach is to factorise the model into smaller components and assume the majority of components are independent. This allows the model then to be parameterised,

$\mathcal{M} = \mathcal{M}(\tau)$. This is typically done with multinomial distributions, which then naturally leads to an implementation based on dynamic Bayesian networks [4]. This approach is explained further in the next section since it underpins expectation propagation methods of parameter optimisation. It should be stressed, however, that the policy optimisation techniques in sections 5 and 6 are applicable to virtually all statistical dialogue systems since they make few assumptions about the underlying dialogue models.

The second key idea of POMDP-based dialogue management is that rather than mapping states into actions as in a conventional system, the policy of a POMDP maps whole belief states into actions. Thus the decision as to what to do next is not simply dependent on a single assumed state, instead it takes account of the full distribution across all states. This mapping can be modelled deterministically as in $a_t = \pi(b_t)$ but since b_t is a high dimension continuous vector, the same tractability issues encountered with the implementation of the dialogue model apply. An alternative approach is to represent the mapping by a stochastic model $\mathcal{P}$

$$a_t \sim \pi(a|b_t; \mathcal{P}). \tag{2}$$

This stochastic model can then be parameterised $\mathcal{P} = \mathcal{P}(\theta)$, which allows a wide range of function approximation techniques to be applied. For example, the features of belief space which directly impact each possible system act a_i can be encoded in a basis function $\phi_{a_i}(b_t)$. A weighted set of these basis functions can then be cast into a probability via a softmax function as in

$$\pi(a_t|b_t; \theta) = \frac{e^{\theta \cdot \phi_{a_t}(b_t)}}{\sum_a e^{\theta \cdot \phi_a(b_t)}} \tag{3}$$

where $\cdot$ denotes a scalar product. Section 5, explains how the parameters θ of this model can be efficiently optimised using a natural actor-critic framework.

However, whereas for slot-based dialogue systems it is often easy make assumptions about the shape of the dialogue model distributions, designing an accurate parameterised model for a policy may be hard. An alternative approach therefore is to use a non-parametric model for the policy. Section 6 explains how Gaussian processes can be used for this purpose.

Unlike conventional classification tasks, the objective of a dialogue system is to achieve some long-term goal through a sequence of planned actions. This is an example of planning under uncertainty and optimisation can be performed using reinforcement learning. Each dialogue turn is assigned a reward r_t based on the current state and action a_t

$$r_t = r(b_t, a_t) = \sum_{s_t} b_t(s_t) r(s_t, a_t). \tag{4}$$

The goal of reinforcement learning is then to maximise the expected discounted value of the total accumulated reward

$$R = \mathcal{E}\left\{ \sum_{t=1}^{T} \gamma^{t-1} r_t \right\}, \tag{5}$$

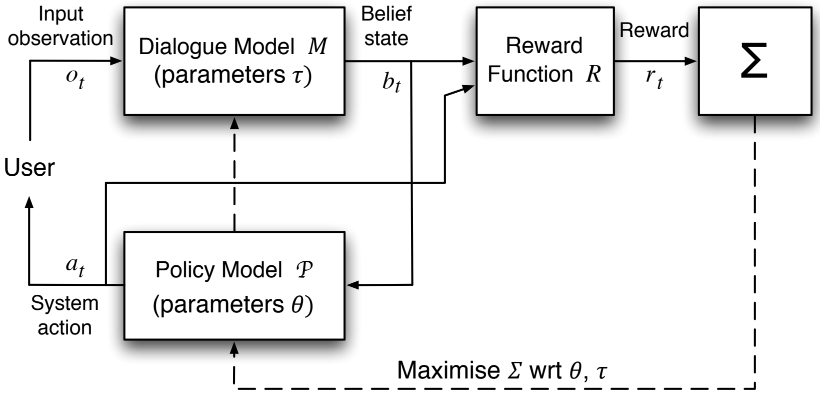

Figure 1: General framework for parameter optimisation in a POMDP-based spoken dialogue system. The input/output interaction may be with a real user or a simulated user.

where T is the length of the dialogue and $\gamma \leq 1$ is a discount factor included to allow the present value of future rewards to be discounted. In practical limited domain spoken dialogue systems such as information inquiry, the reward function will typically have the form

$$r_t = \begin{cases} -1 & \text{if } t < T, \\ +20 & \text{if } t = T \text{ and dialogue successful,} \\ 0 & \text{if } t = T \text{ and dialogue unsuccessful.} \end{cases} \tag{6}$$

In this case, it would be normal to set $\gamma = 1$.

The above sets out the basic framework of a POMDP-based spoken dialogue system and the main elements are summarised in Fig. 1. A dialogue model $\mathcal{M}$ maintains a distribution b_t over all possible dialogue states, updating it each turn in response to each new observation o_t. A dialogue policy $\mathcal{P}$ then determines the best system action a_t to take given the current belief state b_t via a mapping $a_t = \pi(b_t)$. At each turn, a reward r_t is generated and accumulated. The overall objective is to maximise the expected value of this accumulated reward.

Before addressing the issue of how to optimise the dialogue model $\mathcal{M}$ and the policy model $\mathcal{P}$, it will be helpful to give a little more detail on how these models are typically implemented.

3 A Dynamic Bayesian Network Dialogue Model

One strategy that can be used to simplify the dialogue model is to factorise the state into a collection of sub-components. For example, many POMDP dialogue models factorise the state at a particular point in time, s_t, into the user's goal, g_t, the true user action, u_t, and a dialogue history, h_t, as suggested by [16]. In come cases, these sub-components

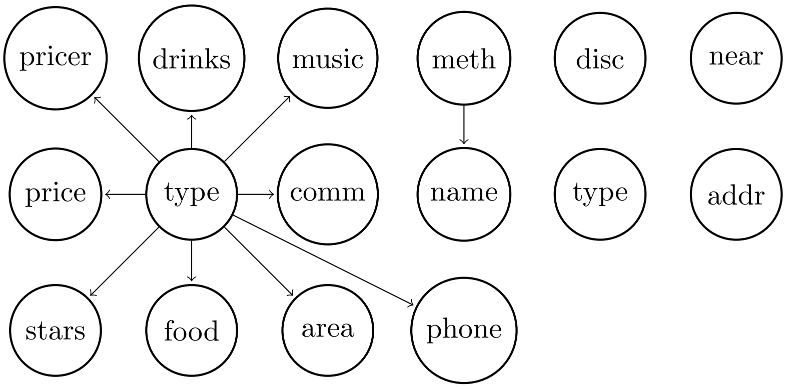

Figure 2: Concepts in the CamInfo restaurant system.

can be further factorised according to a collection of *concepts* $c \in \mathcal{C}$, so that the goal is made up of sub-goals $g_{c,t}$, and similarly $h_t = \{h_{c,t}\}_{c\in\mathcal{C}}$ (see Figure 2).

The CamInfo restaurant system, which will be used for experiments later in the chapter, provides a good example of how this approach can be used in practice. The task of this system is to provide information about restaurants in Cambridge with users speaking to the system over the phone. Figure 2 enumerates the different concepts in the system.

As mentioned in Section 2, the naïve approach of simply implementing equation 1 to update the system's beliefs soon becomes intractable. The factorisation described above, along with some conditional independence assumptions, enables the use of standard machine learning algorithms to update the beliefs. This is done as follows.

First, the variables of interest are represented in a Bayesian network [17]. Each node of the network denotes a random variable of interest and edges in the network encode conditional independence assumptions. The assumption made is that the joint distribution of all nodes factorises into the product of the distribution of each node given its parents in the graph. A Bayesian network representation of part of the CamInfo system is given in Figure 3. In the CamInfo system, the sub-histories are assumed to be dependent on only their previous value, the true user action and the system's action. The observation is dependent on only the user act. The user act depends on all the sub-goals and the last system action while the sub-goals depend on their previous value, the system action and optionally other goals in the network. The dependencies between the goals are shown by the arrows in Figure 2.

The probability functions are grouped into two types. The first type is called the *mostly constant factor* type and it is used for the probabilities of goal nodes and history nodes. Suppose a node is labelled X_t and a collection of "special" values are defined, $x_1, x_2, \ldots, x_k$, along with a collection of "parent classes". Example "special values" are `dontcare` and `N/A` which are used to represent goals where the user doesn't care about the value and where the concept is not applicable. In the CamInfo system, the parent

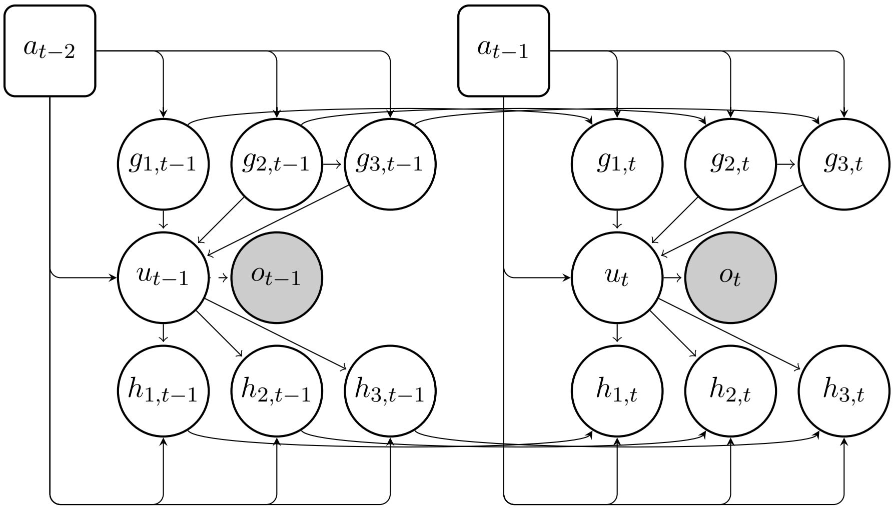

Figure 3: Bayesian network for three concepts in the CamInfo system.

class allows one to define different probability structures for different situations (based on the system action and the node's other parents), but to still allow for significant amounts of parameter tying. An example of such parent classes arises in the case when the system has informed that no venue has a particular concept value. Let the parents of X_t, excluding the node's previous value, be denoted by $par'(X_t)$ and let $\rho(par'(X_t))$ denote the associated parent class. The probabilities are then defined in such a way that the "special values" have distinct probabilities but all remaining values are considered equivalent in order to tie the parameters. Formally, the mostly constant probability functions are defined as follows, with $i, j \in \{1, 2, \ldots, k\}$ and $x, y \notin \{x_1, x_2, \ldots, x_k\}$ and $x \neq y$

$$\begin{aligned}
p(X_t = x_j | X_{t-1} = x_i, \rho(par'(X_t)) = \rho') &= \boldsymbol{\tau}_{\rho',i,j} \\
p(X_t = x | X_{t-1} = x_i, \rho(par'(X_t)) = \rho') &= \boldsymbol{\tau}_{\rho',i,k+1} \\
p(X_t = x_j | X_{t-1} = x, \rho(par'(X_t)) = \rho') &= \boldsymbol{\tau}_{\rho',k+1,j} \\
p(X_t = x | X_{t-1} = x, \rho(par'(X_t)) = \rho') &= \boldsymbol{\tau}_{\rho',i,k+1,k+1} \\
p(X_t = y | X_{t-1} = x, \rho(par'(X_t)) = \rho') &= \boldsymbol{\tau}_{\rho',i,k+1,k+2}.
\end{aligned}$$

This structure of probability factor allows parameters that are expected to be similar to be tied together. For example, in modelling the changes in the user's goal for food, the different types of food will not be included in the list of "special values". As a result, the probability of the user's food goal changing in one turn to the next from, say, Indian to Italian is always equal to the probability of changing between any other pair of food types such as Chinese to French. Similarly, the probability of the user's goal staying Indian will be no different to the probability of the goal staying the same for any other food type.

The second form of probability factor is used to define the probability of the user action given the parent goals and system action. Only two probabilities are differentiated, depending on whether the goals are acceptable pre-conditions for the user action given the system's prompt [18]. For example, if the user has said "I want Chinese food" then goals which include `food=Chinese` are acceptable but goals which have `food=`x for other values of x would not be acceptable. Formally, the probability is defined as

$$p(u|g,a) = \begin{cases} \tau_1 & \text{If g is acceptable as a set of pre-conditions} \\ \tau_2 & \text{otherwise} \end{cases} \tag{7}$$

With the probability network now defined, one can use standard algorithms such as loopy belief propagation to update the beliefs in these variables after every turn[17]. An extension of loopy belief propagation, called expectation propagation, can even provide estimates of the parameters of the model from data [5], and the use of this algorithm for updating and parameter learning will be explained in the next section.

4 Expectation Propagation

Expectation propagation works by approximating the joint distribution as a simpler factorised distribution. Any marginals that are required can then be easily computed by summing out the appropriate variables. The starting point of EP is therefore the joint distribution.

The joint distribution of all variables, X, in the network can be written as a product of probability factors, $p(X) = \prod_f p_f(X)$, with f indexing the factors. Each factor gives the probability of a variable given its parents in the Bayesian network. The collection of variables linked to factor f is denoted X_f. The joint can therefore be written $p(X) = \prod_f p_f(X_f)$. When a collection of variables is observed, the joint posterior distribution is again proportional to $\prod_f p_f(X_f)$, with observed variables replaced by their observed value. Expectation propagation attempts to find an approximation to this posterior, $q(X) = \prod q_f(X_f) \approx p(X)$. In this case, a factorized approximation is used, where each factor is further factorized: $q_f(X_f) = \prod_j q_f(x_j)$, with j indexing all variables and $q_f(x_j)$ constant for variables not appearing in the factor.

EP solves for this approximation one factor at a time. A particular factor, $\tilde{f}$, is chosen and all other factors are fixed. One must then find $q_{\tilde{f}}(X_{\tilde{f}}) = \prod_j q_{\tilde{f}}(x_j)$ to minimize $KL(p||q^{\backslash\tilde{f}} q_f)$, where

$$q^{\backslash\tilde{f}}(X_{\tilde{f}}) \propto \prod_{f \neq \tilde{f}} q_f(X_f).$$

The function $q^{\backslash\tilde{f}}(X_{\tilde{f}})$ denotes the *cavity distribution*, obtained by multiplying all approximations except for $\tilde{f}$. The cavity distribution as a function of a single variable x_j is similarly defined as

$$q^{\backslash\tilde{f}}(x_j) \propto \prod_{f \neq \tilde{f}} q_f(x_j). \tag{8}$$

The function $q^{\backslash\tilde{f}} q_{\tilde{f}}$ is called the *target function.*

In the case here, all the probability factors to be approximated describe the probability of a discrete output variable given a collection of discrete input variables, and a collection of parameter vectors. For clarity, the effect of the parent category, ρ, and the index of the slot, i, will be omitted, and the simplest case of a goal dependent on only its previous value is presented. Note that this exposition uses the general form of probability function instead of the mostly constant probability functions presented in the previous section.

The chosen probability factor, $\tilde{f}$, has the form

$$p(g_t = j | g_{t-1} = k) = \tau_{j,k}$$

and hence approximating functions $q_{\tilde{f}}(g_t)$, $q_{\tilde{f}}(g_{t-1})$, and $q_{\tilde{f}}(\boldsymbol{\tau}_j)$ must be found. All $q_f(\boldsymbol{\tau}_j)$ approximations are constrained to the Dirichlet distribution, with the parameters denoted by $\boldsymbol{\alpha}_{f,j}$. The approximations for the other factors are fixed and the cavity distributions for the variables are defined as per (8). In the case of the discrete variables g_t and g_{t-1}, the cavity distributions are computed by multiplying all factor approximations except for $\tilde{f}$. The cavity distribution for the parameters $\boldsymbol{\tau}_j$ is a product of continuous distributions. For N_f different factors, the cavity distribution is the Dirichlet distribution with parameters

$$\boldsymbol{\alpha}_j^{\backslash\tilde{f}} = \sum_{f \neq \tilde{f}} \boldsymbol{\alpha}_{f,j} - (N_f - 1)\mathbf{1}. \tag{9}$$

Note that when the parameters $\boldsymbol{\tau}_j$ do not appear in a factor, the approximation is constant and the vector of approximation parameters, $\boldsymbol{\alpha}_{f,j}$, equals the vector of ones, $\mathbf{1}$.

Given the cavity distributions, one can show that the discrete approximating functions that minimize $KL(p||q^{\backslash\tilde{f}} q_{\tilde{f}})$ are [19]

$$q_{\tilde{f}}(g_t) \propto \sum_{g_{t-1}} q^{\backslash\tilde{f}}(g_{t-1}) \mathbb{E}(\boldsymbol{\tau}_{g_{t-1},g_t} | q^{\backslash\tilde{f}}(\boldsymbol{\tau}_{g_{t-1}})), \tag{10}$$

$$q_{\tilde{f}}(g_{t-1}) \propto \sum_{g_t} q^{\backslash\tilde{f}}(g_t) \mathbb{E}(\boldsymbol{\tau}_{g_{t-1},g_t} | q^{\backslash\tilde{f}}(\boldsymbol{\tau}_{g_{t-1}})), \tag{11}$$

where the expectations are taken over $q^{\backslash\tilde{f}}$.

It can be shown that to minimize the KL divergence, the set of parameters for the target function, denoted $\boldsymbol{\alpha}_j^*$, must satisfy the following equation for every k [19],

$$\Psi(\alpha_{jk}^*) - \Psi(\sum_{l=1}^{N_\alpha} \alpha_{jl}^*) = c_{jk}, \tag{12}$$

where N_α denotes the number of values,

$$c_{jk} = \Psi(\alpha_{jk}^{\backslash\tilde{f}}) - \Psi(\sum_{l=1}^{N_\alpha} \alpha_{jl}^{\backslash\tilde{f}}) + \frac{w_{jk}}{\alpha_{jk}^{\backslash\tilde{f}}} - \frac{1 - w_{j0}}{\sum_{l=1}^{N_\alpha} \alpha_{jl}^{\backslash\tilde{f}}}, \tag{13}$$

$\Psi(z)$ is the digamma function,

$$\Psi(z) = \frac{d}{dz} \log \Gamma(z), \tag{14}$$

and the w_{jk} are weights ($\sum_k w_{jk} = 1$)

$$w_{j0} \propto \sum_{j' \neq j} q^{\setminus \tilde{f}}(g_{t-1} = j'), \tag{15}$$

$$w_{jk} \propto q^{\setminus \tilde{f}}(g_{t-1} = j) q^{\setminus \tilde{f}}(g_t = k) \frac{\alpha_{jk}^{\setminus \tilde{f}}}{\sum_{l=1}^{N_\alpha} \alpha_{jl}^{\setminus \tilde{f}}}. \tag{16}$$

Various methods are possible for solving equation 12. The approach used here is taken from Section 3.3.3. of [20]. Let $\Delta = \Psi(\sum_{k=1}^{N_\alpha} \alpha_{ik}^*)$, and make α_{ij}^* the subject of the formula in equation 12,

$$\alpha_{jk}^* = \Psi^{-1}(c_{jk} + \Delta). \tag{17}$$

Summing over k and taking both sides as arguments for the Ψ function gives,

$$\Delta = \Psi(\sum_{l=1}^{N_\alpha} \alpha_{jl}^*) = \Psi\left(\sum_{l=1}^{N_\alpha} \Psi^{-1}(c_{jl} + \Delta)\right). \tag{18}$$

One can now solve for Δ using Newton's method and use (17) to obtain the $\boldsymbol{\alpha}_j^*$ parameters. The desired approximating function parameters are then calculated as

$$\boldsymbol{\alpha}_{\tilde{f},j} = \boldsymbol{\alpha}_j^* - \boldsymbol{\alpha}_j^{\setminus \tilde{f}}. \tag{19}$$

The special forms of probability factor described in Section 3 use essentially the same update, though the computation can be simplified because of their special structure. Details can be found in [19]. The full algorithm operates by repeatedly choosing a factor to update, computing the cavity distributions in terms of the current approximations (8) and (9) and then updating the current approximating functions as per (10),(11) and (19). Similar to belief propagation, the process is repeated until changes in the approximating functions fall below a threshold.

5 Policy Gradient Methods

In Section 2, it was noted that some form of approximation is necessary to obtain a tractable policy model, and in the example provided, a stochastic policy was approximated by a softmax function (3) where the policy parameters θ are weights in the linear combination of a set of basis functions ϕ.

One of the most successful approaches to estimating the parameters of such a model depends on the use of policy gradient methods which repeatedly estimate a gradient of the expected reward (5) and take a small step in the gradient direction in order to increase the expected reward. The computation of the gradient is built around a Monte Carlo algorithm which estimates the gradient from a finite number of dialogues generated in interaction with a real user or a user simulator [21]. Under reasonable assumptions, such an approach will converge to a local maximum of the expected reward. The core of a policy gradient algorithm relies on the availability of analytic solutions to computing the gradient of the reward with respect to the policy parameters.

Policy gradient methods are best explained in terms of the complete dialogue history H_t which consists of the observed dialogue history h_t together with the unobserved dialogue states: $H_t = \{s_0, a_0, o_1, s_1, \ldots, a_{t-1}, o_t, s_t\}$. Given this definition, the objective function in (5) can be expressed as the expected reward over all trajectories

$$R(\theta) = \int p(H;\theta)R(H)dH \tag{20}$$

where $R(H)$ is the expected reward accumulated along the complete history H and $p(H;\theta)$ is the probability of the complete history H given the policy parameters θ.

Using "the log likelihood-ratio trick" [22] and Monte Carlo approximation using N sampled dialogues, the gradient can be estimated as follows

$$\nabla R(\theta) \approx \frac{1}{N}\sum_{n=1}^{N} \nabla \log p(H_n;\theta)R(H_n). \tag{21}$$

By definition, the probability $p(H;\theta)$ is the product of the probabilities of all actions, observations and state transitions along the complete history H; therefore, the probability of the complete history is as follows

$$p(H;\theta) = p(s_0)\prod_{t=1}^{T} p(o_t|s_t)p(s_t|a_{t-1}, s_{t-1})\pi(a_{t-1}|b_t;\theta) \tag{22}$$

where $p(s_0)$ is the initial state distribution. Consequently, the log-gradient can be written in the form

$$\nabla \log p(H;\theta) = \sum_{t=0}^{T-1} \nabla \log \pi(a_t|b_t;\theta) \tag{23}$$

since the state observation and transition probabilities do not depend on θ. Substituting this into (21) gives

$$\nabla R(\theta) \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n-1} \nabla \log \pi(a_t^n|b_t^n;\theta)R(H_n). \tag{24}$$

Note that the gradient now depends only on observed variables. To obtain a closed form solution of (24), the policy π must be differentiable with respect to θ. Conveniently, the logarithm of the softmax function (3) is differentiable with respect to θ.

To lower the variance of the estimate of the gradient, a constant baseline, B, can be introduced into (24) without introducing any bias [22]

$$\nabla R(\theta) \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n-1} \nabla \log \pi(a_t^n|b_t^n;\theta)^T (R(H_n) - B). \tag{25}$$

The gradient defined in (25) still cannot be used directly since it includes the expected reward $R(H)$ which is not directly observable. However, it is possible to approximate $R(H)$ by a linear function parameterised by a vector w as follows

$$R(H) \approx R(H;w) = \sum_{t=0}^{T-1} \nabla \log \pi(a_t|b_t;\theta)^T \cdot w + C \tag{26}$$

To compute the parameters w, a least squares method can be used to solve the following set of equations

$$r_n = \sum_{t=0}^{T_n-1} \nabla \log \pi(a_t^n|b_t^n;\theta)^T \cdot w + C \qquad \forall n \in \{1,\ldots,N\} \tag{27}$$

where r_n is the reward observed at the end of each dialogue. Substituting (26) into (25) gives:

$$\nabla R(\theta) \approx \nabla R(\theta;w) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n-1} \nabla \log \pi(a_t^n|b_t^n;\theta) \nabla \log \pi(a_t^n|b_t^n;\theta)^T \cdot w \tag{28}$$

Note that in (28), the baseline B was cancelled by the constant C and it can be shown that C is an optimal constant baseline which minimizes the variance of the gradient [22]. Once the gradient estimate (28) is computed, it can be used to update the policy parameters by: $\theta' \leftarrow \theta + \beta\nabla\, R(\theta;w)$, where β determines the step size.

5.1 Natural Actor Critic algorithm

Although (28) can be used to optimise the policy parameters, the use of the "plain" gradient yields rather poor convergence properties since methods using this gradient often suffer from extremely flat plateaus in the expected reward function. In contrast, the *natural gradient* defined as

$$\widetilde{\nabla}\, R(\theta) = F^{-1}(\theta)\nabla R(\theta), \tag{29}$$

where $F(\theta)$ is the Fisher Information Matrix does not suffer from such behaviour [25]. Based on this idea, a family of Natural Actor Critic (NAC) algorithms which estimates a natural gradient of the expected reward function has been developed [7]. An appealing feature of these algorithms is that in practice the Fisher Information Matrix does not need to be explicitly computed. Inspecting (28), it can be observed that the expression

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n-1} \nabla \log \pi(a_t^n|b_t^n;\theta) \nabla \log \pi(a_t^n|b_t^n;\theta)^T \tag{30}$$

is in fact an estimate of the Fisher Information matrix. Thus, (28) can be written as $\nabla R(\theta) \approx F(\theta)w$ and the natural gradient of the expected reward is simply

$$\widetilde{\nabla} R(\theta) = F^{-1}(\theta)\nabla R(\theta) \approx F^{-1}(\theta)F(\theta)w = w. \tag{31}$$

Hence, the weights which minimise the mean square error in the linear function approximation of $R(H)$ are in fact the components of the natural gradient. A fortunate consequence of this is that use of the natural gradient not only improves performance, but it is also leads to a less computationally intensive algorithm.

Once the parameters w of the approximation of the reward function, $R(H; w)$, have been computed, the policy parameters θ can be iteratively improved by $\theta' \leftarrow \theta + \beta w$.

5.2 Natural Actor and Belief Critic algorithm

In Section 4 it was shown that the Expectation-Propagation algorithm can be used to infer not only the distribution over the unobserved dialogue states but also the dialogue model parameters. The main advantage of the EP algorithm is that since it is unsupervised it does not need annotated data. However, EP does not guarantee a maximization of the expected reward whereas ideally both the dialogue model and the policy parameters should be designed to maximise the expected reward (5). The rest of this section will describe the Natural Actor and Belief Critic (NABC) algorithm which offers a solution to this problem [9].

The NABC algorithm extends policy gradient methods so that the dialogue model parameters are optimised jointly with the policy parameters. Note that this algorithm does not need annotated data; although, information about the rewards received in the sampled dialogues is still necessary.

The stochastic policy given in (3) can be written in more detail to show the dependency of the belief state b_t on the dialogue history h_t and the model parameters τ

$$\pi(a_t|b(\cdot|h_t;\tau);\theta) \approx \frac{e^{\theta^T \cdot \phi_{a_t}(b(\cdot|h_t;\tau))}}{\sum_{\tilde{a}} e^{\theta^T \cdot \phi_{\tilde{a}}(b(\cdot|h_t;\tau))}}. \tag{32}$$

The policy π is now clearly seen to depend on the parameters τ. The difficulty with using policy gradient methods for learning the parameters of the dialogue model $\mathcal{M}(\tau)$ is that since the function ϕ, which extracts features from the belief state, is usually a hand-crafted function of non-continuous features, the policy is not usually differentiable with respect to τ. However, this problem can be alleviated by assuming that the model parameters τ come from a prior distribution $p(\tau; \alpha)$ that is differentiable with respect to the parameters α. Then, this prior can be sampled to give the model parameters during the estimation of the gradient of the expected reward.

The goal of NABC is therefore to learn the parameters α of the prior distribution for the model parameters τ together with the policy parameters θ while maximising the expected reward (5). Let the model parameters τ be sampled from the prior at the beginning of each dialogue, then τ becomes part of the observed history $h_t = \{\tau, a_0, o_1, \ldots, a_{t-1}, o_t\}$. Similarly the complete history H_t is extended for τ to give

$H_t = \{\tau, s_0, a_0, o_1, s_1, \ldots, a_{t-1}, o_t, s_t\}$. Given the formulation above, the expected reward (5) can be written as

$$R(\alpha, \theta) = \int p(H; \alpha, \theta) R(H) dH \tag{33}$$

which depends on both θ and α and its gradient can be estimated by

$$\nabla R(\alpha, \theta) \approx \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(H_n; \alpha, \theta)(R(H_n) - B) \tag{34}$$

where the probability of the trajectory H is defined as

$$p(H; \alpha, \theta) = p(s_0) p(\tau; \alpha) \prod_{t=1}^{T} p(o_t|s_t) p(s_t|a_{t-1}, s_{t-1}) \pi(a_{t-1}|b(\cdot|h_{t-1}; \tau); \theta). \tag{35}$$

Consequently, the log-gradient $p(H; \alpha, \theta)$ has the following form

$$\nabla \log p(H; \alpha, \theta) = \left[\nabla_\alpha \log p(\tau; \alpha)^T, \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|b(\cdot|h_t; \tau); \theta)^T \right]^T. \tag{36}$$

Similar to the derivation of the policy gradient algorithms, this leads to solving the following set of equations using the least squares method

$$r_n = \left[\nabla_\alpha \log p(\tau_n; \alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n; \tau); \theta)^T \right] \cdot [w_\alpha^T, w_\theta^T]^T + C \tag{37}$$

$$\forall n \in \{1, \ldots, N\}$$

where r_n is the reward observed at the end of each dialogue and the solution $[w_\alpha^T, w_\theta^T]^T$ is the natural gradient of the expected reward.

Similar to the Natural Actor Critic algorithm, the $[w_\alpha^T, w_\theta^T]^T$ vector can be used to iteratively improve the policy parameters and the prior of the dialogue model parameters: $\theta' \leftarrow \theta + \beta_\theta w_\theta$, $\alpha' \leftarrow \alpha + \beta_\alpha w_\alpha$. Finally when the estimate of the parameters converges, the dialogue model parameters τ are computed as the expectation of the prior distribution $p(\tau; \alpha)$.

A specific form of the NABC algorithm which updates only the model parameters can also be derived. This method called Natural Belief Critic algorithm can be used not only with stochastic policies but also handcrafted policies [9, 26]. This is especially useful in commercial applications where the behaviour of dialogue systems is constrained by specific and very often handcrafted requirements.

5.3 The dialogue model parameters prior

In order to use NABC in practice, a prior for the model parameters τ is needed. Since the parameters of the Bayesian Network described in Section 3 are parameters of multiple multinomial distributions, a product of Dirichlet distributions provides a convenient prior. Formally, for every node $j \in \{1, \dots, J\}$ in the Bayesian Network, there are parameters τ_j describing a probability $p(j|par(j); \tau_j)$ where the function $par(j)$ defines the parents of the node j. Let $|par(j)|$ be the number of distinct combinations of values of the parents of j. Then, τ_j is composed of the parameters of $|par(j)|$ multinomial distributions and it is structured as follows: $\tau_j = \left[\tau_{j,1}, \dots, \tau_{j,|par(j)|}\right]$. Consequently, a prior for τ_j can be formed from a product of Dirichlet distributions: $\prod_{k=1}^{|par(j)|} Dir(\tau_{j,k}; \alpha_{j,k})$, parameterised by $\alpha_{j,k}$. Let the vector $\tau = [\tau_1, \dots, \tau_J]$ be a vector of all parameters in the Bayesian Network. Then, the probability $p(\tau; \alpha)$ from (37) can be defined as:

$$p(\tau; \alpha) = \prod_{j=1}^{J} \prod_{k=1}^{|par(j)|} Dir(\tau_{j,k}; \alpha_{j,k}), \tag{38}$$

for which a closed form solution of the log-gradient exists and this can be used in (37) to compute the natural gradient of the expected reward.

5.4 Evaluation

To given an indication of performance, an experimental evaluation of the NAC and NABC algorithms was conducted using the BUDS dialogue system as outlined in Section 3 [4]. The BUDS dialogue manager can use both a handcrafted policy and a stochastic policy of the form described in Section 2.

To compare performance of the NAC and NABC algorithms, three dialogue systems were built for the CamInfo Restaurant domain described in Section 3

- **HDC** - a system using a handcrafted dialogue policy and a set of finely tuned handcrafted dialogue model parameters,
- **NAC** - a system using a stochastic policy optimised using the NAC algorithm and a set of handcrafted dialogue model parameters,
- **NABC** - a system using a combined set of stochastic policy and dialogue model parameters jointly optimised using the NABC algorithm.

The systems were trained and tested using an agenda-based user simulator [27] which incorporates a semantic concept confusion model to enable training and testing across a range of semantic error rates. The reward function awards -1 in each dialogue turn and at the end of a dialogue it awards 20 for a successful dialogue and 0 for an unsuccessful one. A dialogue is considered successful if a suitable venue is offered and all further pieces of information are given. In the case where no venue matches the constraints, the dialogue is deemed successful if the system tells the user that no venue matches and a suitable alternative is offered. Since a typical dialogue will require around five or six turns to complete, this implies that around 15 represents an upper bound on the achievable mean reward.

The systems were trained by executing 120 iterations with the simulator set to produce error rates between 0% and 50%, uniformly distributed among the dialogues. The total number of sampled dialogues per iteration was 32k. In total, 881 parameters were estimated for the policy $\mathcal{P}$ and 577 for the model $\mathcal{M}$. Both the policy parameters and the parameters of the prior of the dialogue model were initialised by uninformative (uniform) parameters.

Figure 4 compares the learning curves of the NAC and the NABC systems. In the beginning, the NAC algorithm learns faster as it does not have to learn the model parameters; however, as more iterations of training are completed, the performance of the fully trainable system outperforms the baseline with the handcrafted policy. After 120 iterations, both the model and the policy parameters converge.

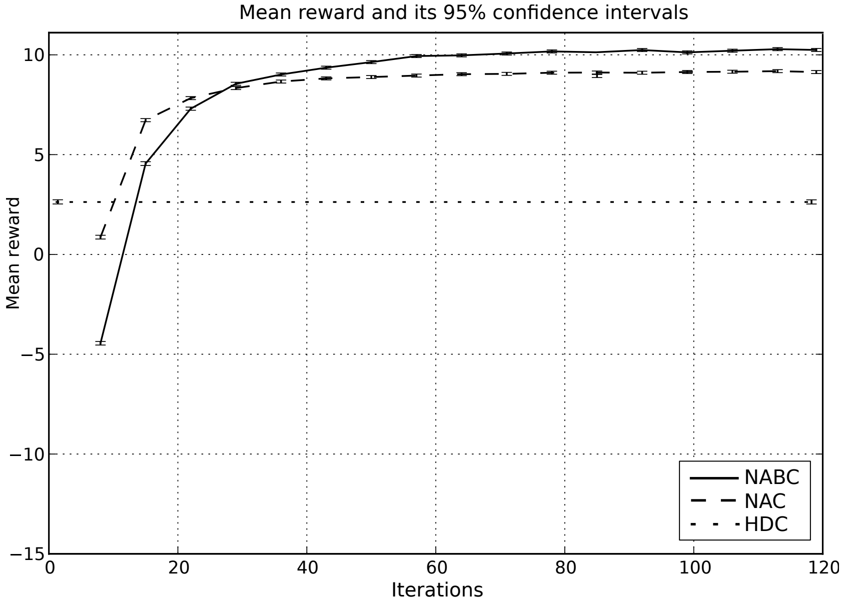

Figure 4: Average reward across all error rates during training of the NAC and NABC systems. The horizontal line gives the performance of the handcrafted system (HDC).

The performance of the HDC, NAC, and NABC systems plotted as a function of the semantic error rate is depicted in Figure 5 At each error rate, 5000 dialogues were simulated and to reduce the variance of results, the training and evaluation procedures were executed 5 times and the results averaged. The results show that the NAC algorithm is very efficient in optimizing the policy parameters. For example, at 35% error rate, the mean reward was increased from -1.96 to 7.35. The results also show that the additional optimization of the dialogue model parameters, performed by joint optimization of the policy and the dialogue model parameters using the NABC algorithm, significantly improves the performance. For example, at 35% error rate, the mean reward was improved by 23% from 7.35 to 9.03.

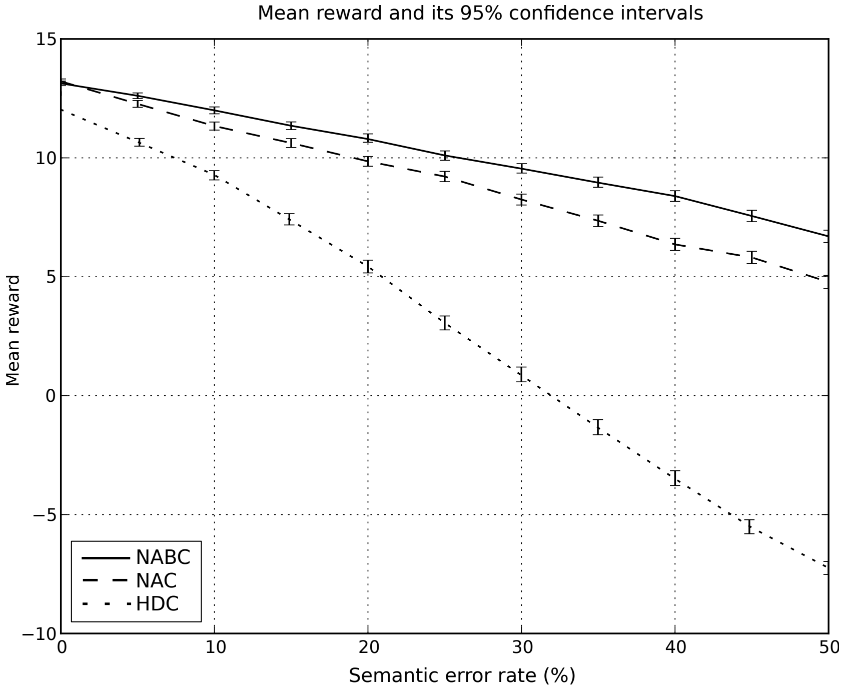

Figure 5: Comparison of the HDC, NAC and NABC systems. The graph plots the mean reward against the semantic error rate.

6 Gaussian Processes for Policy Modelling

As mentioned in the introduction, the aim of a dialogue policy is to map each belief state into an optimal action, i.e. the action that leads to the highest reward at the end of the dialogue. However, exact solutions to policy optimisation are only tractable for problems with very small state-action spaces. Parametric approximations, such as the softmax stochastic policy described in section 5, offer tractable solutions but typically require the basis functions that extract the key features of the belief state to be handcrafted. Furthermore, gradient-based optimisation such as NAC requires large numbers of dialogues to form accurate estimates of the gradient and even then, they can only be optimal subject to the approximation inherent in the selected basis functions.

Alternative methods of policy optimisation discretise the belief space into grid points. This allows the use of standard reinforcement learning algorithms to optimise the policy. However, in order to make real-world dialogue tasks tractable, the belief space must be compressed into a so-called summary space which leads to similar approximation issues. Furthermore, even when compressed to a few hundred grid points, around 10^5 dialogues are still required for optimisation [28]. This number is too large to allow training with real users so the interaction is normally performed with a simulated user. This raises further issues relating to both the cost and complexity of building an adequate simulator, and the degree to which such a simulator can truly reflect real human user behaviour.

In the light of the above, this section briefly describes a rather different approach to policy modelling and optimisation using Gaussian processes to provide non-parametric Bayesian models for function approximation. An advantage of Bayesian approaches is that they offer a principled way of incorporating prior knowledge about the underlying task into the learning process, which gives them the potential to improve learning rates. It is important that the dependencies in the different parts of the belief state space are taken into consideration during learning. Gaussian processes are able to incorporate the prior knowledge of these dependencies elegantly through the choice of a so-called *kernel function*, the purpose of which is to describe correlations in different parts of the space. In addition, Gaussian processes allow the variance of the posterior to be estimated, thus modelling the uncertainty of the approximation. This is particularly useful for dialogue management, since for every belief state-action pair the Gaussian process not only provides a policy estimate, but it also provides a measure of the uncertainty inherent in taking that action in that belief state.

The next section explains how Gaussian processes can be used for policy modelling. In Section 6.2 the core idea of the method is explained on a toy dialogue problem, where different aspects of GP learning are examined and the results are compared. Section 6.3 then demonstrates how this methodology can be applied to a real-world dialogue problem.

6.1 Gaussian Process Reinforcement Learning

The role of a dialogue policy π is to map each summary state $\mathbf{b}$ into a summary action a so as to maximise the expected discounted cumulative reward, which is defined by the Q-function

$$Q(\mathbf{b}, a) = \max_{\pi} E_{\pi} \left(\sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r_{\tau} | \mathbf{b}_t = \mathbf{b}, a_t = a \right) \tag{39}$$

where r_τ is the reward obtained at time τ, T is the dialogue length and γ is the discount factor, $0 < \gamma \leq 1$.

The problem of obtaining the optimal policy is thus equivalent to the problem of obtaining the optimal Q-function

$$\pi(\mathbf{b}) = \arg\max_{a} Q(\mathbf{b}, a). \tag{40}$$

In this case both the policy and its associated Q-function are deterministic. However, both can be modelled as stochastic processes. Firstly, the Q-function can be modelled as a zero mean Gaussian process by providing a kernel function which defines the correlations in different parts of the belief state and action spaces

$$Q(\mathbf{b}, a) \sim \mathcal{GP}\left(0, k((\mathbf{b}, a), (\mathbf{b}, a))\right). \tag{41}$$

The kernel $k(\cdot, \cdot)$ is often factored into separate kernels over the belief state and action spaces $k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}) k_{\mathcal{A}}(a, a)$.

Given this prior and some observed belief state-action pairs, the posterior of the Q-function can be computed. For a sequence of belief state-action pairs $\mathbf{B}_t = [(\mathbf{b}^0, a^0), \ldots, (\mathbf{b}^t, a^t)]^\mathsf{T}$ visited in a dialogue and the corresponding observed immediate rewards $\mathbf{r}_t = [r^1, \ldots, r^t]^\mathsf{T}$, the posterior of the Q-function for any belief state-action pair $(\mathbf{b}, a)$ is defined by

$$Q(\mathbf{b}, a)|\mathbf{r}_t, \mathbf{B}_t \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), cov((\mathbf{b}, a), (\mathbf{b}, a))) \tag{42}$$

where

$$\begin{aligned}
&\overline{Q}(\mathbf{b}, a) = \mathbf{k}_t(\mathbf{b}, a)^\mathsf{T}\mathbf{H}_t^\mathsf{T}(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\mathsf{T} + \sigma^2\mathbf{H}_t\mathbf{H}_t^\mathsf{T})^{-1}\mathbf{r}_t,\\
&cov((\mathbf{b}, a), (\mathbf{b}, a)) = k((\mathbf{b}, a), (\mathbf{b}, a))\\
&\quad -\mathbf{k}_t(\mathbf{b}, a)^\mathsf{T}\mathbf{H}_t^\mathsf{T}(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\mathsf{T} + \sigma^2\mathbf{H}_t\mathbf{H}_t^\mathsf{T})^{-1}\mathbf{H}_t\mathbf{k}_t(\mathbf{b}, a),\\
&\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix},\\
&\mathbf{K}_t = [\mathbf{k}_t((\mathbf{b}^0, a^0)), \ldots, \mathbf{k}_t((\mathbf{b}^t, a^t))],\\
&\mathbf{k}_t(\mathbf{b}, a) = [k((\mathbf{b}^0, a^0), (\mathbf{b}, a)), \ldots, k((\mathbf{b}^t, a^t), (\mathbf{b}, a))]^\mathsf{T}
\end{aligned} \tag{43}$$

and where σ^2 is a measure of the assumed additive noise in the estimate of the Q-function and γ is the discount factor. Matrix $\mathbf{H}_t$ captures the discounting of the reward in Eq. 39 (see [30, 15] for details). Matrix $\mathbf{K}_t$, also called the *Gram matrix*, defines the correlations between the data points.

The marginal likelihood of the observed rewards has an analytic solution

$$\mathbf{r}_t|\mathbf{B}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_t(\mathbf{K}_t + \sigma^2\mathbf{I})\mathbf{H}_t^\mathsf{T}), \tag{44}$$

and this provides the relation between the observed rewards and the kernel function via the Gram matrix.

The kernel function can be parameterised in the form $k(\cdot, \cdot) = k(\cdot, \cdot; \Theta)$ where Θ are the kernel parameters, also called the *hyper-parameters*. Note that these parameters are different to the policy parameters described earlier in this chapter. The main difference is that the way these parameters are set does not restrict the optimality of the solution, instead their effect is mainly on the number of data points needed and the resulting accuracy of the optimal solution. If the kernel function is parameterised, the Gram matrix is also parameterised $\mathbf{K}_t = \mathbf{K}_t(\Theta)$. Given a corpus of visited belief states, with the actions taken and resulting rewards, the hyper-parameters can be estimated by maximising the marginal likelihood from (44).

Due to the matrix inversion in (42), the computational complexity of calculating the Q-function posterior is $O(t^3)$, where t is the number of data points. In the case of a dialogue system, the number of points used for estimation will be equal to the total number of turns, summed over all dialogues and this number can be very large. For this reason, a sparse approximation is needed which can ensure that all the data points are taken into account whilst reducing the computational complexity. One such method is the kernel span sparsification method [31]. The basic idea of this method is to select a small subset of data points, the *representative points*, with which the kernel function can be effectively approximated. It can be shown that this reduces the complexity of calculating the posterior to $O(tm^2)$, where m is the number of representative points.

The description so far has only given the stochastic model for the Q-function. However, what is required for the dialogue management is the model for the policy π. One way of defining a policy is via an ϵ-greedy approach. It requires setting of an additional parameter ϵ which balances how often the action is taken based on the current best estimate of the Q-function mean—the exploration—and how often an action is taken randomly—the exploration. Thus, the model becomes

$$a = \begin{cases} \arg\max_a \overline{Q}(\mathbf{b}, a) & \text{with probability } 1 - \epsilon \\ \mathit{random} & \text{with probability } \epsilon \end{cases} \tag{45}$$

Alternatively, active learning may be incorporated into the action selection process in order to provide efficient data selection [32, 33]. The main idea behind active learning is to select only the data points that contribute the most to the estimate. Here, similar to [34], active learning can be used for more efficient exploration. During exploration, actions are chosen based on the variance of the GP estimate for the Q-function and during exploitation, actions are chosen based on the mean

$$a = \begin{cases} \arg\max_a \overline{Q}(\mathbf{b}, a) & \text{with probability } 1 - \epsilon \\ \arg\max_a \mathit{cov}\left((\mathbf{b}, a), (\mathbf{b}, a)\right) & \text{with probability } \epsilon \end{cases} \tag{46}$$

Both of these approaches require a manual balancing of exploration and exploitation by tuning the parameter ϵ. Since the Gaussian process for the Q-function defines a Gaussian distribution for every summary state-action pair (Eq. 42), when a new summary point $\mathbf{b}$ is encountered, for each action $a_i \in \mathcal{A}$, there is a Gaussian distribution $Q(\mathbf{b}, a_i) \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a_i), \mathit{cov}((\mathbf{b}, a_i), (\mathbf{b}, a_i))))$. Sampling from these Gaussian distributions gives a set of Q-values for each action $\{Q^i(\mathbf{b}, a_i)\}$ from which the action with the highest sampled Q-value can be selected

$$a = \arg\max_{a_i} Q^i(\mathbf{b}, a_i). \tag{47}$$

Thus, this approach maps the GP approximation of the Q-function into a stochastic policy model.

All of these approaches allow observations to be processed sequentially, in direct interaction with the user, whether real or simulated. A specific and commonly used algorithm for achieving this is GP-Sarsa which is particularly suited to online episodic reinforcement learning [31, 30, 15].

6.2 Gaussian Process Reinforcement Learning for a simple Voice Mail Dialogue Task

The practical application of the above ideas to a spoken dialogue task and the use of the GP-Sarsa algorithm will first be illustrated using a very simply ”toy” voice mail dialogue system [35]. This system has just three states: the user asked for the message either to be saved or deleted, or the dialogue ended; and three actions: ask the user what to do, save or delete the message. The observation of what the user actually wants to do at each turn is corrupted with noise. For both learning and evaluation, a simulated user is used which generates an observation error with probability 0.3 and terminates the dialogue after at most 10 turns. In the final state, the system receives a reward of 10 for operating as intended or a penalty of -100 otherwise, eg because it deleted rather than saving the mail. Each intermediate state receives the penalty of -1. In order to keep the problem simple, a model defining the transition and observation probabilities is assumed so that the belief can be easily updated, but policy optimisation is performed on-line.

The kernel function must accurately represent prior knowledge about the Q-function correlations and it must be defined for both states and actions. Since the action space is discrete, a simple δ kernel can be defined over actions

$$k(a,a') = 1 - \delta_a(a'), \tag{48}$$

where δ_a is the Kronecker delta function.

In contrast, the state space is continuous space and it is assumed that points in belief space which are in some sense similar will be more strongly correlated. Here four different kernel functions are investigated, listed in Table 6.2. Each kernel function defines a different correlation. The polynomial kernel views elements of the state vector as features, the dot-product of which defines the correlation. They can be given different relevance r_i in the parameterised version. The Gaussian kernel accounts for smoothness, *i.e.,* if two states are close to each other the Q-function in these states is correlated. The scaled norm kernel defines positive correlations in the points that are close to each other and a negative correlation otherwise. This is particularly useful for the voice mail problem, where, if two belief states are very different, taking the same action in these states generates a negatively correlated reward.

kernel function	expression
polynomial	$k(\mathbf{x},\mathbf{x}') = \langle \mathbf{x},\mathbf{x}' \rangle$
parameterised poly.	$k(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{D} \frac{x_i x_i'}{r_i^2}$
Gaussian	$k(\mathbf{x},\mathbf{x}') = p^2 \exp \frac{-\lVert \mathbf{x}-\mathbf{x}' \rVert^2}{2\sigma_k^2}$
scaled norm	$k(\mathbf{x},\mathbf{x}') = 1 - \frac{\lVert \mathbf{x}-\mathbf{x}' \rVert^2}{\lVert \mathbf{x} \rVert^2 \lVert \mathbf{x}' \rVert^2}$

Table 1: Kernel functions

Since this toy problem is very simple, it is possible to compute an exact solution for the optimal policy, for example using the POMDP solver toolkit [37]. Hence, the policy learnt by GP-Sarsa can be compared with the exact solution, and in order to assess the efficiency with which it learns, it can also be compared with a standard grid-based algorithm such as the Monte Carlo Control (MCC) algorithm [36, 13].

The dialogue manager was therefore trained in interaction with the simulated user and the performance was compared between the grid-based MCC algorithm and GP-Sarsa using the different kernel functions from Table 6.2. Kernel hyper-parameters were optimised using 300 sample dialogues obtained using the exact optimal policy. All the algorithms use an ϵ-greedy approach where the exploration rate ϵ was fixed at 0.1. In order to reduce the affects of statistical variation, for every training set-up, exactly the same training iterations were performed using 1000 different random generator seedings. After every 20 dialogues the resulting 1000 partially optimised policies were evaluated by testing it on 1000 dialogues.

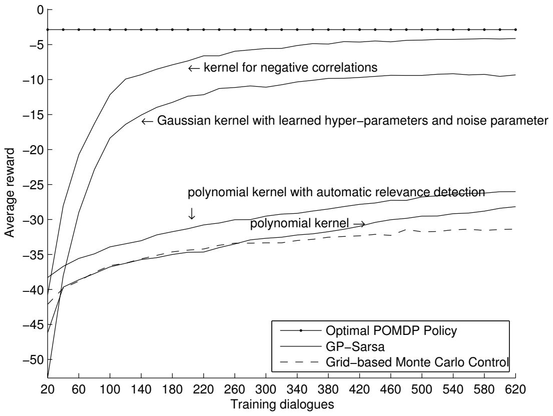

Figure 6: Evaluation results on Voice Mail task

The results are shown in Fig. 6. As can be seen, the grid-Based MCC algorithm has a relatively slow convergence rate. GP-Sarsa with the polynomial kernel exhibited a learning rate similar to MCC in the first 300 training dialogues, continuing with a more upward learning trend. The parameterised polynomial kernel performs slightly better. The Gaussian kernel, however, achieves a much faster learning rate. The scaled norm kernel achieved close to optimal performance in 400 dialogues, with a much higher convergence rate then the other methods. Thus, GP-Sarsa can learn close to optimal policies much more quickly than standard reinforcement learning algorithms but the choice of kernel is clearly very important.

6.3 Gaussian Process Reinforcement Learning for a Real-world Tourist Information Task

To show that the GP approach can be scaled to practical dialogue systems, this section will briefly describe its application to policy optimisation in the Cambridge CamInfo tourist information system which provides information about restaurants, bars, hotels, and other tourist attractions in the Cambridge area. The database consists of more than 400 entities each of which has up to 10 attributes that the user can query.

The CamInfo system can be configured to run with either the BUDS based dialogue manager described in Section 3 or with the Hidden Information State (HIS) dialogue manager [13]. Since the HIS system is slightly easier to adapt to GP-Sarsa, that version will be described here.

The summary state in the HIS system is a four-dimensional space consisting of two elements that are continuous (the probability of the top two hypotheses) and two discrete elements (one relating the portion of the database entries that matches the top partition and the other relating to the last user action type). The summary action space is discrete and consists of eleven elements. The nature of the HIS state space is quite different from that of the toy problem and kernels that have negative correlations, such as the scaled norm kernel cannot be used. Instead, and somewhat contrary to the results for the toy problem, a second order polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^2$ turns out to be the most appropriate for the continuous elements. For the discrete elements, the δ-kernel (6.2) is used.

As previously, policy optimisation is performed by interacting with a simulated user and the system receives a reward of 20 or 0, depending on whether or not the dialogue was successful, less the number of turns taken to fulfil the user's request. Fig. 7 shows a graph of performance as a function of the number of dialogues used for policy optimisation for both GP-Sarsa and the baseline grid-based MCC algorithm.

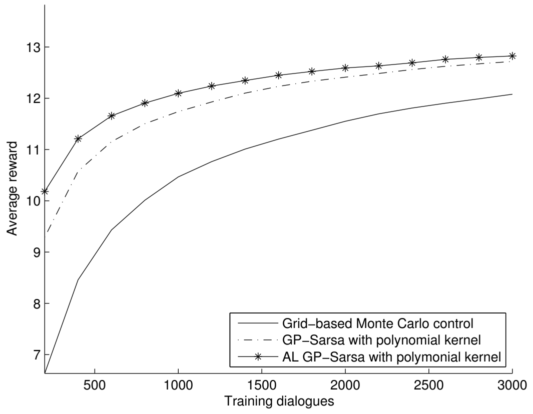

Figure 7: Evaluation results on CamInfo task

The results show that in the very early stage of learning, *i.e.,* during the first 400 dialogues, the GP-based method learns much faster. Furthermore, the learning process can be accelerated by using active learning (AL) where the actions are selected based on the estimated uncertainty as in (46). After performing many iterations both the GP-Sarsa and the grid-based MCC algorithms converge to the same performance.

Fig. 7 shows that active learning using an ϵ-greedy policy with variance exploration learns faster than a standard random exploration. However, this graph only displays the average reward. If GP is to be used to learn on-line with real users then the variance of

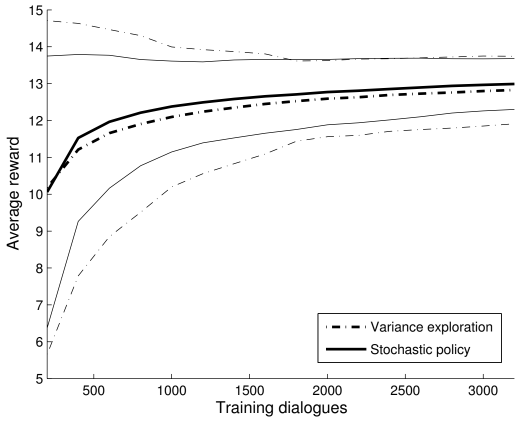

Figure 8: Average reward vs training epoch with a simulated user for a stochastic policy compared to an ϵ-greedy policy with variance exploration. The upper and lower bounds in each case denote 95% confidence intervals.

the reward is also a concern.

Fig. 8 compares the learning rates of an ϵ-greedy policy with variance exploration with that of the fully stochastic policy given by (47). As can be seen the learning rates are similar, but the variance of the stochastic policy is much reduced. This suggests that for practical systems the use of the stochastic policy is preferable because it reduces the risk of taking bad actions during learning. When real users are involved, this benefit may well be significant.

7 Conclusions

Statistical spoken dialogue systems based on the framework of partially observable Markov decision processes (POMDPs) offer considerable potential for reducing costs and increasing robustness. However, practical systems have complex probability models and very large state spaces. The availability of efficient algorithms for optimising model parameters and policies is therefore essential for developing these systems further.

This chapter has described three different approaches to optimisation. Firstly, it was shown that by incorporating the model parameters into a Bayesian Network representation of the dialogue model and using an extended form of inference algorithm called Expectation-Propagation (EP), it is possible to learn model parameters on-line in parallel with belief monitoring. Secondly, it was shown that standard policy gradient methods are not only effective for policy exploration, they can also be used to optimise model parameters. This is particularly appealing because unlike EP, the gradient approach directly maximises the expected reward. Finally, the use of a non-parametric approach was presented in which the Q function is modelled as a Gaussian Process. The key features of this approach are that it includes an explicit model of state space correlations and it explicitly provides a measure of the uncertainty in the current Q function estimate. These allow the GP approach to achieve much faster learning and safer exploration of alternative actions.

References

[1] N Roy, J Pineau, and S Thrun. Spoken Dialogue Management Using Probabilistic Reasoning. In *Proceedings of the ACL 2000*, 2000.

[2] SJ Young. Talking to Machines (Statistically Speaking). In *Int Conf Spoken Language Processing*, Denver, Colorado, 2002.

[3] JD Williams and SJ Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):393–422, 2007.

[4] B Thomson and SJ Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588, 2010.

[5] TP Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proc 17th Conf in Uncertainty in Artificial Intelligence*, pages 362–369, Seattle, 2001. Morgan-Kaufmann.

[6] B Thomson, F Jurcicek, M Gasic, S Keizer, F Mairesse, K Yu, and SJ Young. Parameter learning for POMDP spoken dialogue models. In *IEEE Workshop on Spoken Language Technology (SLT 2010)*, Berkeley, CA, 2010.

[7] J Peters and S Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[8] B Thomson, J Schatzmann, and SJ Young. Bayesian Update of Dialogue State for Robust Dialogue Systems. In *Int Conf Acoustics Speech and Signal Processing ICASSP*, Las Vegas, 2008.

[9] F Jurcicek, B Thomson, and SJ Young. Natural Actor and Belief Critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs. *ACM Transactions on Speech and Language Processing*, 7(3), 2011.

[10] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[11] E Engel, S Mannor, and R Meir. Reinforcement learning with Gaussian processes. In *ICML 2005*, Bonn, Germany, 2005.

[12] M Gasic, F Jurcicek, S Keizer, F Mairesse, B Thomson, K Yu, and SJ Young. Gaussian Processes for Fast Policy Optimisation of a POMDP Dialogue Manager for a Real-world Task. In *SigDial 2010*, Tokyo, Japan, 2010.

[13] SJ Young, M Gasic, S Keizer, F Mairesse, J Schatzmann, B Thomson, and K Yu. The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174, 2010.

[14] M Gašić and S Young. Effective Handling of Dialogue State in the Hidden Information State POMDP Dialogue Manager. *ACM Transactions on Speech and Language Processing*, To appear.

[15] M Gašić. *Statistical dialogue modelling.* PhD thesis, University of Cambridge, 2011.

[16] Jason D. Williams, Pascal Poupart, and Steve Young. Factored partially observable Markov decision processes for dialogue management. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialog Systems*, 2005.

[17] Chris Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[18] S. Keizer, M. Gasic, F. Mairesse, B. Thomson, K. Yu, and S. Young. Modelling user behaviour in the HIS-POMDP dialogue manager. In *Proceedings of SLT*, pages 121–124, 2008.

[19] B. Thomson. *Statistical methods for spoken dialogue management.* PhD thesis, University of Cambridge, 2009.

[20] U. Paquet. *Bayesian inference for latent variable models.* PhD thesis, University of Cambridge, 2007.

[21] J. Schatzmann. *Statistical User and Error Modelling for Spoken Dialogue Systems.* PhD thesis, University of Cambridge, 2008.

[22] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 1992.

[23] R.S. Sutton, D. McAllester, S. Singh, and Y Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[24] V. Konda and J.N. Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 2000.

[25] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[26] F Jurčíček, B Thomson, S Keizer, F Mairesse, M Gašić, K Yu, and SJ Young. Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *Proc. Interspeech*, pages 90–93. ISCA, 2010.

[27] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young. Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE ASRU '05: Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, 2005.

[28] M Gašić, S Keizer, F Mairesse, J Schatzmann, B Thomson, K Yu, and S Young. Training and evaluation of the HIS-POMDP dialogue system in noise. In *Proceedings of SIGDIAL*, 2008.

[29] M Gašić, F. Lefèvre, F Jurčíček, S Keizer, F Mairesse, B Thomson, K Yu, and S Young. Back-off Action Selection in Summary Space-Based POMDP Dialogue Systems. In *Proceedings of ASRU*, 2009.

[30] Y Engel. *Algorithms and Representations for Reinforcement Learning.* PhD thesis, Hebrew University, 2005.

[31] Y Engel, S Mannor, and R Meir. Reinforcement learning with Gaussian processes. In *Proceedings of ICML*, 2005.

[32] MacKay, D.J.C. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[33] D Cohn, L Atlas, and R Ladner. Improving Generalization with Active Learning. *Machine Learning*, 15:201–221, 1994.

[34] MP Deisenroth, CE Rasmussen, and J Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

[35] JD Williams. *Partially Observable Markov Decision Processes for Spoken Dialogue Management.* PhD thesis, University of Cambridge, 2006.

[36] RS Sutton and AG Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, Massachusetts, 1998.

[37] AR Cassandra. POMDP solver, 2005.

[38] CE Rasmussen and CKI Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, Massachusetts, 2005.