POLICY OPTIMISATION OF POMDP-BASED DIALOGUE SYSTEMS WITHOUT STATE SPACE COMPRESSION

M. Gašić, M. Henderson, B. Thomson, P. Tsiakoulis and S. Young

Cambridge University Engineering Department
Trumpington St, Cambridge, CB2 1PZ, UK

ABSTRACT

The partially observable Markov decision process (POMDP) has been proposed as a dialogue model that enables automatic improvement of the dialogue policy and robustness to speech understanding errors. It requires, however, a large number of dialogues to train the dialogue policy. Gaussian processes (GP) have recently been applied to POMDP dialogue management optimisation showing an ability to substantially increase the speed of learning. Here, we investigate this further using the Bayesian Update of Dialogue State dialogue manager. We show that it is possible to apply Gaussian processes directly to the belief state, removing the need for a parametric policy representation. In addition, the resulting policy learns significantly faster while maintaining operational performance.

Index Terms— Gaussian process, POMDP, statistical dialogue modelling

1. INTRODUCTION

The partially observable Markov decision process (POMDP) has been proposed as a dialogue model which intrinsically deals with the uncertainty from the speech recogniser. It thus has the potential to provide a more robust operation of the system [1, 2, 3, 4]. It assumes that the dialogue state s_t is only partially observable and depends on a noisy observation o_t which is defined by the *observation probability* $P(o_t|s_t)$. Since the dialogue state is unobservable, at every dialogue step t a distribution over all states is maintained which is called the *belief state* b_t. It takes values $\mathbf{b} \in \mathcal{B}$, where $\mathcal{B}$ is a continuous space of dimensionality $|\mathcal{S}|$, namely $[0,1]^{|\mathcal{S}|}$. The dialogue policy then maps the belief state $\mathbf{b}$ into an appropriate action a, $\pi(\mathbf{b}) = a$. Exact updating of the belief state requires a summation over all states which rapidly becomes intractable for very large state spaces:

$$\begin{aligned} b(s_{t+1} = s') \quad & \propto P(o_{t+1} = o'|s_{t+1} = s') \\ & \textstyle\sum_{s \in \mathcal{S}} P(s_{t+1} = s'|a_t = a, s_t = s) b(s_t = s). \end{aligned} \tag{1}$$

However, there exist real-world dialogue systems based on the POMDP model which maintain an approximate belief state in real-time throughout the dialogue [5, 4].

Exact policy optimisation for POMDPs is also intractable for everything but very simple cases [6]. Moreover, approximate POMDP solutions, as in the point-based value iteration algorithm [7, 8], are only suitable for relatively small action/state problems. However, a POMDP with discrete state and observation spaces can be viewed as an MDP with a continuous state space [6]. This allows MDP reinforcement learning algorithms to be used for the policy optimisation and with further approximations achieves tractability.

To achieve tractability two approximations are typically used. The first involves compressing the belief state space into a summary space where the policy can be tractably optimised and then heuristically mapped back into the original space [9, 10, 4, 5]. The second approximation assumes a parametric representation of the policy [11, 4, 12].

Compressing the belief state into a summary space is non-trivial and often requires knowledge about the dialogue domain. If the summary space is too large the performance of the optimised policy can degrade [13] or the learning process may significantly slow down [12]. Methods for automatic selection of summary space features do exist, but they have not been applied directly to the POMDP belief space [14, 15]. An alternative method described in [16] automatically summarises information from the spoken language understanding component and then builds a POMDP using that summarised space to ensure tractability in learning. However, this still requires heuristics and it does not facilitate the incorporation of prior knowledge regarding the task domain and user behaviour.

Stated more generally, any parametric representation of the policy limits the solution to be optimal only within the chosen basis. Furthermore, the basis needs to be predefined by the designer, often requiring knowledge about the domain. Avoiding the need for summary space mapping or parametric policy representation is therefore extremely desirable.

This paper explores ways of overcoming the above issues through the use of Gaussian processes (GPs), which are non-parametric models for Bayesian inference. We show using the example of the Bayesian Update of Dialogue State (BUDS) system that it is possible to directly apply GP policy optimisation to the belief space. This achieves similar policy performance to standard methods and reduces the required training data by an order of magnitude whilst avoiding the need for hand-crafting a summary space mapping.

In the next section, we provide an overview of Gaussian process POMDP policy optimisation. Section 3 gives a description of the BUDS dialogue manager. Then, in Section 4, the domain is described and the training and evaluation results are presented. Section 5 concludes the paper with directions for future research.

2. GAUSSIAN PROCESSES IN POMDP POLICY OPTIMISATION

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

$$Q(\mathbf{b}, a) = E_\pi \left(\sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r_\tau | b_t = \mathbf{b}, a_t = a \right), \tag{2}$$

where r_τ is the reward obtained at time τ, T is the dialogue length and γ is the discount factor, $0 < \gamma \leq 1$. Therefore, optimising the Q-function is equivalent to optimising the policy π.

A Gaussian process (GP) is a non-parametric Bayesian probabilistic model that can be used for function regression [17]. It is fully defined by a mean and a kernel function. The kernel function defines prior function correlations and is crucial for obtaining good posterior estimates with just a few observations. The Q-function can thus be modelled as a Gaussian process, $Q(\mathbf{b}, a) \sim \mathcal{GP}\left(0, k((\mathbf{b}, a), (\mathbf{b}, a))\right)$ where the kernel $k(\cdot,\cdot)$ is factored into separate kernels over the belief state and action spaces [18]:

$$k(\mathbf{b}, a, \mathbf{b}', a') = k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}')k_{\mathcal{A}}(a, a'). \tag{3}$$

In addition, if we assume additive noise in the Q-function, $\Delta Q(\mathbf{b}, a) \sim \mathcal{N}(0, \sigma^2)$, the following recursive stochastic relationship between a reward and the Q-function can be established:

$$\begin{aligned} r_{t+1}(b(s_t) = \mathbf{b}, a_t = a) &= Q(\mathbf{b}, a) - \gamma Q(\mathbf{b}', a') \\ &+ \Delta Q(\mathbf{b}, a) - \gamma \Delta Q(\mathbf{b}', a'), \end{aligned} \tag{4}$$

where $b(s_{t+1}) = \mathbf{b}'$ is the next belief state and $a' = \pi(\mathbf{b}')$ is the next action, $a_{t+1} = a'$.

This relationship, together with the assumed Gaussian process prior and for a sequence of observed rewards, allow for the Q-function posterior to be calculated [19]. More specifically, for a sequence of belief state-action pairs $\mathbf{B}_t = [(\mathbf{b}^0, a^0), \ldots, (\mathbf{b}^t, a^t)]$ visited in a dialogue and the corresponding observed immediate rewards $\mathbf{r}_t = [r^1, \ldots, r^t]^\intercal$, the posterior of the Q-function for any belief state-action pair $(\mathbf{b}, a)$ is a Gaussian distribution:

$$Q(\mathbf{b}, a)|\mathbf{r}_t, \mathbf{B}_t \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), \mathrm{cov}((\mathbf{b}, a), (\mathbf{b}, a))), \tag{5}$$

where the posterior mean and covariance are given by:

$$\begin{aligned} &\overline{Q}(\mathbf{b}, a) = \mathbf{k}_t(\mathbf{b}, a)^\intercal \mathbf{H}_t^\intercal (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\intercal + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\intercal)^{-1} \mathbf{r}_t, \\ &\mathrm{cov}((\mathbf{b}, a), (\mathbf{b}, a)) = k((\mathbf{b}, a), (\mathbf{b}, a)) \\ &\quad - \mathbf{k}_t(\mathbf{b}, a)^\intercal \mathbf{H}_t^\intercal (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\intercal + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\intercal)^{-1} \mathbf{H}_t \mathbf{k}_t(\mathbf{b}, a) \end{aligned} \tag{6}$$

such that matrix $\mathbf{H}_t$ defines the linear relationship expressed in Eq. 4:

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix}, \tag{7}$$

and $\mathbf{K}_t$ is the Gram matrix over visited belief state-action pairs:

$$\begin{aligned} &\mathbf{K}_t = [\mathbf{k}_t((\mathbf{b}^0, a^0)), \ldots, \mathbf{k}_t((\mathbf{b}^t, a^t))], \\ &\mathbf{k}_t(\mathbf{b}, a) = [k((\mathbf{b}^0, a^0), (\mathbf{b}, a)), \ldots, k((\mathbf{b}^t, a^t), (\mathbf{b}, a))]^\intercal, \end{aligned} \tag{8}$$

It can be shown [20] that the marginal likelihood of the observed rewards is modelled by

$$\mathbf{r}_t|\mathbf{B}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_t(\mathbf{K}_t + \sigma^2 \mathbf{I})\mathbf{H}_t^\intercal). \tag{9}$$

Due to the matrix inversion in Eq. 6, the computational complexity of calculating the 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. This poses a serious computational problem. We therefore use the kernel span sparsification method described in [19] to reduce the computational complexity.

The kernel span sparsification method approximates the kernel function using a subset of visited belief state-action pairs called the dictionary $\mathcal{D} = \{(\tilde{\mathbf{b}}^0, \tilde{a}^0), \ldots, (\tilde{\mathbf{b}}^m, \tilde{a}^m)\}$. It requires placing a threshold ν on the precision to which the kernel function is calculated and thus it allows the dictionary size to be kept small:

$$\min_{\mathbf{g}_t} \left(k((\mathbf{b}^t, a^t), (\mathbf{b}^t, a^t)) - \tilde{\mathbf{k}}_{t-1}(\mathbf{b}^t, a^t)^\intercal \mathbf{g}_t \right) \leq \nu, \tag{10}$$

where $\tilde{\mathbf{k}}_{t-1}(\mathbf{b}^t, a^t) = [k((\mathbf{b}^t, a^t), (\tilde{\mathbf{b}}^0, \tilde{a}^0)), \ldots, k((\mathbf{b}^t, a^t), (\tilde{\mathbf{b}}^m, \tilde{a}^m))]^\intercal$ and $\mathbf{g}_t$ is a vector of coefficients. Every time the threshold ν is exceded a new point is added to dictionary. In this way, the Gram matrix is approximated as $\mathbf{K}_t \approx \mathbf{G}_t \tilde{\mathbf{K}}_t \mathbf{G}_t^\intercal$, where $\mathbf{G}_t = [\mathbf{g}_1, \ldots, \mathbf{g}_t]$ and $\tilde{\mathbf{K}}_t$ is the Gram matrix over the dictionary points. This reduces the computational complexity to $O(tm^2)$, where m is the number of dictionary points. In practice, the threshold ν represents the trade-off between the accuracy of the assumed prior correlations and the computational complexity.

Given a set of belief state-action pairs $\mathbf{B}_t$ and associated rewards $\mathbf{r}_t$, the above approximation of the posterior Q-function defines a Gaussian distribution for every belief state-action pair (Eq. 5). Thus, when a new belief point $\mathbf{b}$ is encountered, for each action $a \in \mathcal{A}$, there is a Gaussian distribution $Q(\mathbf{b}, a) \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), \mathrm{cov}((\mathbf{b}, a), (\mathbf{b}, a))))$. Sampling from these Gaussian distributions gives a set of Q-values for each action $\{Q(\mathbf{b}, a) : a \in \mathcal{A}\}$ from which the action with the highest sampled Q-value can be selected:

$$\pi(\mathbf{b}) = \arg\max_a \{Q(\mathbf{b}, a) : a \in \mathcal{A}\}. \tag{11}$$

Thus, the GP approximation of the Q-function is effectively transformed into a stochastic policy model, which can be optimised to maximise the reward. Here we use it in combination with ϵ-greedy learning where ϵ was fixed at 0.1 to prevent the policy reaching a local optimum too early during the training. GP-Sarsa has been shown to be effective for on-line optimisation [18]. Furthermore, it can be amended for episodic reinforcement learning and hence applied to dialogue optimisation [20].

3. BAYESIAN UPDATE OF DIALOGUE STATE DIALOGUE MANAGER

The Bayesian Update of Dialogue State dialogue manager is a POMDP-based dialogue manager. Its belief state consists of the marginal posterior probability distribution for each of the hidden nodes of a dynamic Bayesian network representing the dialogue state [21].

In order to find the optimal policy the following approach is normally taken. The belief space $\mathcal{B}$ is mapped into a discrete summary space $\mathcal{C}$. This summary space, also referred to as the grid space, is then mapped into a feature space $\mathcal{F}$. Likewise, the action space is mapped into a smaller scale summary action space. The feature space is used to produce a parametric representation of the policy and the Q-function. A weighted set of these basis functions can then be cast into a probability via a soft-max function as in

$$\pi(a|\mathbf{c}; \theta) = \frac{e^{\theta \cdot f_a(\mathbf{c})}}{\sum_a e^{\theta \cdot f_a(\mathbf{c})}}. \tag{12}$$

where $\mathbf{c} \in \mathcal{C}$, $f_a(\mathbf{c})$ are the features for action a, and θ are the policy parameters. The policy is then optimised using gradient methods and the natural actor critic (NAC) algorithm has been found to be effective for this purpose [22, 21]. Once a summary action has been selected, it is heuristically mapped back into a system action using information from the belief space.

Mappings $\mathcal{B} \to \mathcal{C}$ and $\mathcal{C} \to \mathcal{F}$ require a significant amount of hand-crafting. Also, due to the parametric policy representation, the solution is only optimal within the chosen basis. Finally, gradient methods are known to be inherently slow, which effectively prohibits direct on-line optimisation with real users.

The experiments which follow show that by using GP-Sarsa it is possible to optimise a policy in the summary space $\mathcal{C}$ much more efficiently than by using NAC. Even more importantly, it is also possible to define and optimise a policy using the full belief space $\mathcal{B}$ thereby avoiding the need for hand-crafted summary space mappings altogether.

4. EXPERIMENTAL EVALUATION

This section presents experimental results for a spoken dialogue system working in the restaurant domain. The GP-Sarsa algorithm is compared to the NAC algorithm both in terms of the speed of learning and the performance of the resulting policy, evaluated using both a simulated user and a real user trial.

4.1. The Application Domain

The application domain consists of all restaurants in Cambridge, UK that are listed in the TopTable web service [23]. There are about 150 such restaurants each described by 8 attributes such as *food-type*, *price-range*, *location*, etc. This data is compiled into a Bayesian network representation of the belief space using an ontology which defines relationship between the dialogue concepts. The network comprises of 25 hidden nodes and each represents a distribution over 3 to 150 values. The summary space is formed from 200 hand-crafted binary features and the action space consists of 16 summary actions. More details of this application are given in [24].

For training and simulation testing, an agenda-based user simulator was used [25]. The reward function was set to give a reward of 20 for successful dialogues, zero otherwise, minus the number of dialogue turns. The dialogue length is limited to 30 turns.

4.2. The GP kernel function

The kernel function in the case of the summary state system is a product of kernels over the summary state space and the discrete action space, as defined in Eq. 3. For the summary state space a simple linear inner product was used in the form

$$k_{\mathcal{C}}(\mathbf{c}, \mathbf{c}') = \langle \mathbf{c}, \mathbf{c}' \rangle + 1 \tag{13}$$

and for discrete actions a simple δ-kernel

$$k_{\mathcal{A}}(a, a') = 1 - \delta_a(a') \tag{14}$$

where $\delta_a(a') = 1$ iff $a = a'$.

For the full-space systems, the kernel function is a product of kernels over the belief state and the discrete action space (Eq. 3). The kernel function over the belief state was constructed from the sum of individual kernels over the distribution $\mathbf{b}_k$ for each hidden node k in the Bayesian network belief state $\mathbf{b}$. The kernel function of two corresponding hidden nodes was based on the expected likelihood kernel [26], which is also a simple linear inner product:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}') = \sum_k \langle \mathbf{b}_k, \mathbf{b}'_k \rangle, \tag{15}$$

where $\mathbf{b}_k$ is the probability distribution encoded in the kth hidden node.

The hidden nodes in the BUDS system are divided into the history nodes and the goal nodes for each concept in the dialogue, eg. *area*, *food-type*, *address*.

The history nodes define possible dialogue histories for a particular concept, eg. *system-informed*, *user-informed*. For this class of nodes, the kernel of two history nodes is a simple inner product between the corresponding node distributions.

The goal nodes define possible values for a particular concept, eg. *Chinese*, *Indian*. While it is possible to calculate the kernel function for the goal nodes in the same way as for the history nodes, in this case, the choice of system action, such as *confirm* or *inform*, does not depend on the actual values. It rather depends on the shape of the distribution and, in particular, it depends on the probability of the most likely value compared to the rest. Therefore, to exploit the correlations further, the kernel over two goal nodes is calculated as the dot product of vectors, where each vector represent the corresponding distribution sorted into order of probability. The only exceptions are the goal for the *method* node and the *discourse act* node. The former defines whether the user is searching for a venue *by name* or *by constraints* and the latter defines which discourse act the user used, eg. *acknowledgement*, *thank you*. Their kernels are calculated in the same way as for the history nodes.

The kernel over actions for the full-space system is the same as for the summary space system (Eq. 14).

As noted in the introduction, the sparsification threshold ν approximates the kernel function with a fixed number of dictionary points. Based on intermediate experimentation to achieve a compact set of dictionary points whilst maintaining good performance, the sparsification threshold was fixed at 0.1 for the summary space kernel and 0.001 for the full belief space kernel.

The choice of the noise σ depends on the reward function. A good rule of thumb is that σ should be the square root of half the range of reward function values. The reward interval as defined in the previous section is $[-30, 20]$ so σ was fixed at 5 for all systems.

4.3. Policy Learning Rate

One of the main motivations for using Gaussian Processes in policy optimisation is to speed up the learning process. Therefore, we first examine the learning rates of different policy models.

The baseline policy (NAC-F) (Eq. 12) was trained using the natural actor-critic algorithm in feature space $\mathcal{F}$ and compared to the GP stochastic policy (Eq. 11) both in the summary space $\mathcal{C}$ (GP-C) and in the full belief space $\mathcal{B}$ (GP-B). All three training procedures used exactly the same number of dialogues with the user simulator that produces noisy output in a form of a 10-best list. The noise error rate is measured as the percentage of the time the correct hypothesis is not within the list of alternatives generated by the simulator error model. All training procedures had the error rate fixed at 15%. Note, however, that the 1-best error rate is actually higher. On the other hand, zero error rate means that no noise is inserted, so the true hypothesis is the only hypothesis given to the dialogue manager.

To measure the rate at which each algorithm learns, the following procedure was adopted. After every 5000 dialogues the partially optimised policy was evaluated on 1000 dialogues at zero error rate.

The averaged reward and average number of turns per dialogue are shown in Figs. 1 and 2 respectively, (success rates follow a similar trend to the reward rates). The GP based policies reach comparable performance to the fully trained NAC policy in approximately $10,000$ dialogues, reducing the required training corpus size by an

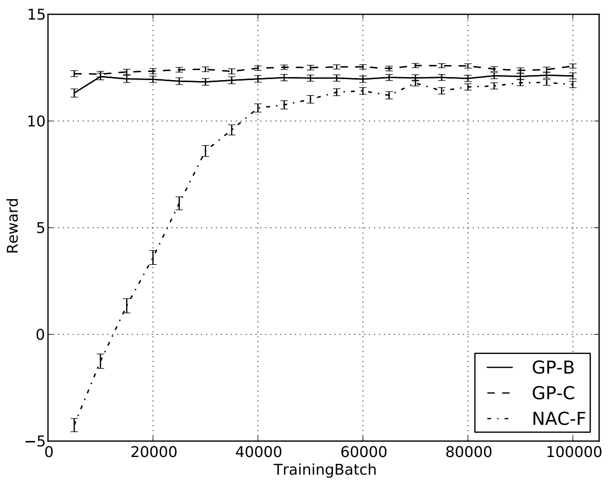

Fig. 1. Learning rates for NAC and GP in summary space, and GP in full space: reward vs training iteration.

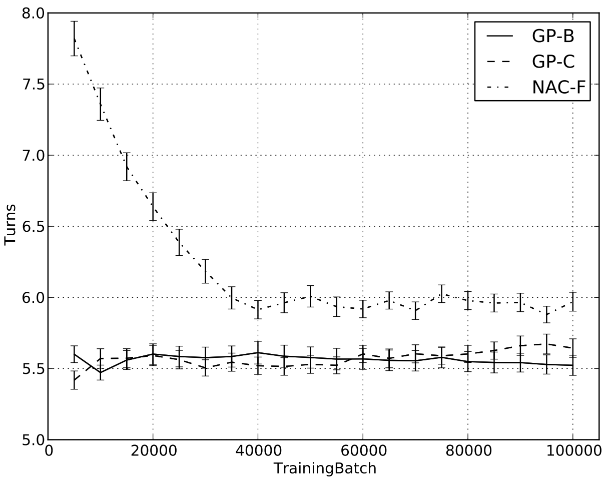

Fig. 2. Learning rates for NAC and GP in summary space, and GP in full space: turns vs training iteration.

order of magnitude. In addition, they produce significantly shorter dialogues. Further, the GP-B policy outperforms the NAC-F policy, whilst the GP-C policy outperforms both. This suggests that parametric policy modelling limits the optimality of the solution and that it is possible to optimise the policy directly on the full belief space. Note that increasing the dimensionality of the input space from summary to full belief space did not significantly reduce the learning rate. The GP-Sarsa algorithm is able to deal with large input spaces by making use of the correlations defined by the kernel function, which in practice makes it effectively independent of the dimensionality of the input space. On the other hand, the summary space $\mathcal{C}$ relies on the incorporation of global dialogue features such as whether the system previously informed the user about a particular entity or that there is no matching venue in the database. This can contribute to its better performance but has all the drawbacks of hand-crafting.

4.4. Qualitative Policy Comparison

To gain some insights into the characteristics of the different methods of optimisation, four policies were compared using a user simulator: the three fully trained policies described above (trained on $100,000$ dialogues), NAC-F, GP-C and GP-B plus GP-B-PART which is similar to GP-B but trained on only $10,000$ dialogues. The systems were tested at zero error rates by performing 1000 dialogues.

To examine the differences between the policies, the summary actions were divided into four broad groups:

- actions where the system *informs* the user, either about a particular slot or about an entity.
- actions where the system asks the user to *select* between two values, e.g. Are you looking for cheap or moderate price range?
- actions which ask for *confirmations*.
- actions which *request* more information from the user, e.g. What kind of food would you like?

The percentage of times each policy uses these actions are given in Table 1. It can be seen that the NAC policy confirms and requests more often than the GP policies. Conversely, the GP policies have learned to use the select action more often, presumably because it is often more useful to explicitly ask the user to choose between two values when the user asks for something and then later changes their mind rather than continuing to confirm the original value which the user no longer wants. In terms of the differences between the policies operating on the full belief space to the policies operating on the summary space, it can be seen that the summary space-based policies request more often and inform less often. Sometimes a degree of guessing about what the user is looking for can be more effective than explicitly requesting all necessary information before accessing the database, especially if the system is not aware that the user has already been informed about a particular entity. Whilst this shows the ability of the full belief space model to exploit the behaviour patterns of the simulated user, it also signals the potential dangers of over-training on the simulated user to the extent that the policy attempts to exploit traits in the behaviour of the user simulator which are not actually present in the real user population.

Table 1. Analysis of actions the policies take in interaction with the user simulator by frequency of use in the four major categories of system action: inform, select, confirm and request.

	GP-B	GP-B-PART	GP-C	NAC-F
Inform	71.2%	75.2%	58.7%	59.1%
Select	2.9%	1.6%	4.0%	0.9%
Confirm	0.3%	0.0%	0.2%	2.3%
Request	25.7%	23.2%	37.1%	37.7%

4.5. Real User Trial

In order to examine the differences between the GP-Sarsa policies and the NAC policy with real users, we conducted an evaluation using the Amazon Mechanical Turk service in a similar set-up to that described in [27]. The trial subjects were assigned specific tasks in the TopTable restaurant domain for Cambridge in which they were asked to find restaurants that had particular features. To elicit more complex dialogues, subjects were sometimes given constraints which might not be satisfied in which case they were given alternative constraints to try. For example, if tasked to find a Vietnamese restaurant in the east side of the city and there were no such restaurants, then the subject might be asked to find a Chinese restaurant instead. Due to the negotiative-style of the dialogues, it is difficult to measure the objective success accurately. Therefore, after each dialogue, subjects were asked to fill in a feedback form indicating whether they thought the dialogue was successful or not (subjective success).

Table 2. Results of the user trial: number of dialogues per policy (#N), subjective success [with confidence interval], average number of user turns per dialogue (with standard error) and average word error rate (WER).

	#N	Success	Ave User Turns	WER
NAC-F	252	94.4 $[90.9, 96.9]$	7.4 (0.2)	22.0
GP-C	249	95.2 $[91.7, 97.5]$	6.7 (0.2)	19.9
GP-B-PART	249	91.6 $[87.4, 94.7]$	7.0 (0.3)	22.4
GP-B	265	93.6 $[89.9, 96.2]$	7.0 (0.2)	21.7

The number of dialogues collected for each policy, the average success rate based on the user rating with confidence intervals, the average dialogue length together with the standard error and the average word error rate are given in Table 2.

The results show that while the success rates between dialogues generated by different policies are not statistically different, the GP-Sarsa policies produced significantly shorter dialogues than the NAC policy. Since shorter dialogues result in higher rewards, this is consistent with the simulation results which showed higher rewards for the GP policies. It is also important to note that the performance of the policy trained with only $10,000$ dialogues (GP-B-PART) was comparable to the policy trained with $100,000$ dialogues (GP-B) suggesting that it should be possible to train a GP-Sarsa policy on the full belief space directly in interaction with real users.

Finally, as in section 4.4, an analysis was performed of the actions taken by the four policies in the user trial (see Table 3). It can be seen that the distributions of actions are very different to the ones taken in interaction with the simulated user (Table 1), suggesting that real users do behave differently to the simulated user. It is interesting to note that the differences between GP-B-PART and GP-B are more prominent on real users. This suggests that while it may be possible to train an effective policy with only $10,000$ dialogues, it is very important that the training conditions match the trial conditions.

5. CONCLUSION

This paper has described how Gaussian processes can be applied to POMDP-based dialogue management. Using the GP-Sarsa algorithm it has been shown that not only is GP policy optimisation faster by an order of magnitude compared to natural gradient methods, it can also be applied directly to the full belief space obviating the need for hand-crafted summary space mappings whilst maintaining both speed of learning and operational performance.

Table 3. Analysis of actions the policies take in the real user trial by frequency of use in the four major categories: inform, select, confirm and request.

	GP-B	GP-B-PART	GP-C	NAC-F
Inform	78.7%	69.1%	72.1%	69.6%
Select	2.6%	1.8%	8.3%	4.0%
Confirm	1.3%	4.6%	1.2%	2.0%
Request	17.4%	24.5%	18.4%	24.4%

Future work will focus on three research directions. The first will investigate training a GP-Sarsa policy operating on the full belief state space in direct interaction with real users. Policies operating on the full belief space should in principle be capable of significantly outperforming summary space policies especially if the summary space omits features that are not present in the belief space. However, this is hard to demonstrate when training on a user simulator, due to differences in the behaviour of real users in real noise conditions.

The second thread of research concerns defining the kernel function and in particular the noise variance σ. It has been shown on toy dialogue problems that the kernel function can be parametrised and its parameters can be learnt off-line directly from data [28, 20], so it remains to apply this method to a real-world dialogue manager.

Finally, while the work presented here obviates the need to reduce the belief state space to a summary space, it still requires the use of a summary action space that must be predefined by the designer. For the experiments here, a simple δ-kernel was used over the summary action space. Gaussian process reinforcement learning will allow more elaborate kernel functions to be defined on the action space. Therefore, in principle it should be possible to define a kernel over the full action space and this needs further investigation.

6. ACKNOWLEDGEMENTS

This work was supported by PARLANCE (www.parlance-project.eu), an EU Seventh Framework Programme project (grant number 287615).

7. REFERENCES

[1] N Roy, J Pineau, and S Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of ACL*, 2000.

[2] B Zhang, Q Cai, J Mao, E Chang, and B Guo, "Spoken Dialogue Management as Planning and Acting under Uncertainty," in *Proceedings of Eurospeech*, 2001.

[3] SJ Young, "Talking to Machines (Statistically Speaking)," in *Proceedings of ICSLP*, 2002.

[4] B Thomson, *Statistical methods for spoken dialogue management*, Ph.D. thesis, University of Cambridge, 2009.

[5] S Young, M Gašić, 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*, vol. 24, no. 2, pp. 150–174, 2010.

[6] LP Kaelbling, ML Littman, and AR Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[7] J Pineau, G Gordon, and S Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of IJCAI*, 2003, pp. 1025–1032.

[8] JD Williams and SJ Young, "Scaling POMDPs for Spoken Dialog Management," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2116–2129, 2007.

[9] JD Williams and SJ Young, "Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI)," in *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, 2006.

[10] JD Williams and SJ Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[11] J Henderson, O Lemon, and K Georgila, "Hybrid Reinforcement/Supervised Learning for Dialogue Policies from fixed data sets," *Computational Linguistics*, vol. 34, no. 4, pp. 487–511, 2008.

[12] L Daubigney, M Geist, and O Pietquin, "Off-policy Learning in Large-scale POMDP-based Dialogue Systems," in *Proceedings of ICASSP*, 2012.

[13] 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.

[14] F Lefèvre, M Gašić, S Keizer, F Mairesse, B Thomson, K Yu, and S Young, "k-nearest neighhbor Monte-Carlo control algorithm form POMDP-based dialogue systems," in *Proceedings of SIGDIAL*, 2009.

[15] L Li, JD Williams, and S Balakrishnan, "Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection," in *Proceedings of Interspeech*, 2009.

[16] F Pinault and F Lefèvre, "Semantic graph clustering for POMDP-based spoken dialogue systems," in *Proceedings of Interspeech*, 2011.

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

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

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

[20] M Gašić, *Statistical Dialogue Modelling*, PhD thesis, University of Cambridge, 2011.

[21] B Thomson and S Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech and Language*, vol. 24, no. 4, pp. 562–588, 2010.

[22] J Peters and S Schaal, "Natural Actor-Critic," *Neurocomputing*, vol. 71, pp. 1180–1190, 2008.

[23] TopTable, "TopTable," 2012, *https://www.toptable.com*.

[24] P Tsiakoulis, M Gašić, M Henderson, J Planells-Lerma, J Prombonas, B Thomson, K Yu, S Young, and E Tzirkel, "Statistical Methods for Building Robust Spoken Dialogue Systems in an Automobile," in *Proceedings of the 4th Applied Human Factors and Ergonomics*, 2012.

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

[26] T Jebara, R Kondor, and A Howard, "Probability product kernels," *J. Mach. Learn. Res.*, vol. 5, pp. 819–844, Dec. 2004.

[27] F Jurčíček, S Keizer, M Gašić, F Mairesse, B Thomson, K Yu, and S Young, "Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk," in *Proceedings of Interspeech*, 2011.

[28] M Gašić, F Jurčíček, S Keizer, F Mairesse, J Schatzmann, B Thomson, K Yu, and S Young, "Gaussian Processes for Fast Policy Optimisation of POMDP-based Dialogue Managers," in *Proceedings of SIGDIAL*, 2010.