POMDP-based dialogue manager adaptation to extended domains

M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis and S. Young
Cambridge University Engineering Department
{mg436,cb404,mh521,dk449,mos25,brmt2,pt344,sjy}@eng.cam.ac.uk

Abstract

Existing spoken dialogue systems are typically designed to operate in a static and well-defined domain, and are not well suited to tasks in which the concepts and values change dynamically. To handle dynamically changing domains, techniques will be needed to transfer and reuse existing dialogue policies and rapidly adapt them using a small number of dialogues in the new domain. As a first step in this direction, this paper addresses the problem of automatically extending a dialogue system to include a new previously unseen concept (or slot) which can be then used as a search constraint in an information query. The paper shows that in the context of Gaussian process POMDP optimisation, a domain can be extended through a simple expansion of the kernel and then rapidly adapted. As well as being much quicker, adaptation rather than retraining from scratch is shown to avoid subjecting users to unacceptably poor performance during the learning stage.

1 Introduction

Existing spoken dialogue systems are typically designed to operate in a static and well-defined domain, and are not well suited to tasks in which the concepts and values change dynamically. For example, consider a spoken dialogue system installed in a car, which is designed to provide information about nearby hotels and restaurants. In this case, not only will the data change as the car moves around, but the concepts (or slots) that a user might wish to use to frame a query will also change. For example, a restaurant system designed to be used within cities might not have the concept of 'al fresco' dining and could not therefore handle a query such as "Find me a French restaurant where I can eat outside". In order to make this possible, techniques will be needed to extend and adapt existing dialogue policies.

Adaptation can be viewed as a process of improving action selection in a different condition to the one in which the policy was originally trained. While adaptation has been extensively studied in speech recognition (see an overview in (Gales and Young, 2007)), in spoken dialogue systems it is still relatively novel and covers a wide range of possible research topics (Litman and Pan, 1999; Litman and Pan, 2002; Georgila and Lemon, 2004; Janarthanam and Lemon, 2010).

A recent trend in statistical dialogue modelling has been to model dialogue as a partially observable Markov decision process (POMDP). This provides increased robustness to errors in speech understanding and automatic dialogue policy optimisation via reinforcement learning (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2007; Young et al., 2010; Thomson and Young, 2010). A POMDP-based dialogue manager maintains a distribution over every possible dialogue state at every dialogue turn. This is called the *belief state*. Based on that distribution the system chooses the action that gives the highest expected reward, measured by the Q-function. The Q-function for a belief state and an action is the expected cumulative reward that can be obtained if that action is taken in that belief state. The optimisation typically requires $\mathcal{O}(10^5)$ to $\mathcal{O}(10^6)$ dialogues, so is normally done in interaction with a simulated user (Jurčíček et al., 2011b).

In reinforcement learning, policy adaptation has been addressed in the context of *transfer learning* (Taylor and Stone, 2009). The core idea is to exploit expertise gained in one domain (source domain) to improve learning in another domain (target domain). A number of techniques have been developed but they have not been previously applied to dialogue management.

Proceedings of the SIGDIAL 2013 Conference, pages 214–222,
Metz, France, 22-24 August 2013. ©2013 Association for Computational Linguistics

Gaussian process (GP) based reinforcement learning (Engel, 2005) has been recently applied to POMDP dialogue policy optimisation in order to exploit the correlations between different belief states and thus reduce the number of dialogues needed for the learning process (Gašić et al., 2010).

An important feature of a Gaussian process is that it can incorporate a prior mean and variance for the function it estimates, in this case the Q-function. Setting these appropriately can significantly speed up the process of learning. If the mean or the variance are estimated in one environment, for example a particular user type or a particular domain, they can be used as a prior for adaptation in a different environment, i.e. another user type or another domain. A Gaussian process does not depend on the belief state but on the correlation between two belief states encoded by the *kernel function*. Therefore, if one defines a kernel function for two belief states in one domain, the policy can be used in a different domain, provided that the correlations between belief states follow a similar pattern.

This paper explores the problem of extending an existing domain by introducing a previously unseen slot. Specifically, a simple restaurant system is considered which allows a user to search for restaurants based on *food-type* and *area*. This domain is then extended by introducing an additional *price-range* slot. The policy is trained for the basic two-slot domain and then reused in the extended domain by defining a modified kernel function and using adaptation. This strategy not only allows for the knowledge of a previously trained policy to be reused but it also guards against poor performance in the early stages of learning. This is particularly useful in a real-world situation where the adaptation is performed in direct interaction with users. In addition, a potential application of this technique to reduce the number of training dialogues is examined. The domain is decomposed into a series of simple domains and the policy is gradually adapted to the final domain with a smaller number of dialogues than are normally needed for training.

The rest of the paper is organised as follows. In Section 2 the background on Gaussian processes in POMDP optimisation is given. Then Section 3 gives a description of the Bayesian Update of Dialogue State dialogue manager, which is used as a test-bed for the experiments. In Section 4, a simple method of kernel modification is described which allows a policy trained in the basic domain to be used in an extended domain. Methods of fast adaptation are investigated in Section 5 and this adaptation strategy is then tested via interaction with humans using the Amazon Mechanical Turk service in Section 6. Finally, the use of repeated adaptation to speed up the process of policy optimisation by learning gradually from simple to more complex domains is explored in Section 7, before presenting conclusions in Section 8.

2 Gaussian processes in POMDPs

The role of a dialogue policy π is to map each belief state $\mathbf{b} \in \mathcal{B}$ into an action $a \in \mathcal{A}$ so as to maximise the expected cumulative reward, a measure of how good the dialogue is.

The expected cumulative reward is defined by the Q-function as:

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

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

A Gaussian process (GP) is a non-parametric Bayesian probabilistic model that can be used for function regression (Rasmussen and Williams, 2005). It is fully defined by a mean and a kernel function which defines prior function correlations.

GP-Sarsa is an on-line RL algorithm that models the Q-function as a Gaussian process (Engel et al., 2005), $Q(\mathbf{b}, a) \sim \mathcal{GP}(0, k((\mathbf{b}, a), (\mathbf{b}, a)))$ where the kernel $k(\cdot, \cdot)$ is factored into separate kernels over the belief state and action spaces $k_\mathcal{C}(\mathbf{b}, \mathbf{b}')k_\mathcal{A}(a, a')$. 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 the following:

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

where $\mathbf{K}_t$ is the Gram matrix – the matrix of the kernel function values for visited points $\mathbf{B}_t$, $\mathbf{H}_t$ is a linear operator that captures the reward lookahead from the Q-function (see Eq. 1) and σ^2 is an additive noise parameter which controls how much variability in the Q-function estimate we expect during the process of learning.

If we assume that the Gaussian process places a prior mean on the Q-function, $Q(\mathbf{b},a) \sim \mathcal{GP}\left(m(\mathbf{b},a), k((\mathbf{b},a),(\mathbf{b},a))\right)$ then the posterior mean $\overline{Q}(\mathbf{b},a)$ is given by (Rasmussen and Williams, 2005):

$$
\overline{Q}(\mathbf{b},a) = m(\mathbf{b},a) + \mathbf{k}_t(\mathbf{b},a)^{\mathsf{T}}\mathbf{H}_t^{\mathsf{T}}(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^{\mathsf{T}} + \sigma^2\mathbf{H}_t\mathbf{H}_t^{\mathsf{T}})^{-1}(\mathbf{r}_t - \mathbf{m}_t), \tag{3}
$$

where $\mathbf{m}_t = [m(\mathbf{b}^0,a^0),\ldots,m(\mathbf{b}^t,a^t)]^{\mathsf{T}}$. The estimate of the variance is same as in Eq. 2.

The Q-function posterior in Eqs. 2 and 3 defines a Gaussian distribution for every belief state-action pair. Thus, when a new belief state $\mathbf{b}$ is encountered, for each action $a \in \mathcal{A}$, there is a Gaussian distribution $Q(\mathbf{b},a) \sim \mathcal{N}(\overline{Q}(\mathbf{b},a), cov((\mathbf{b},a),(\mathbf{b},a))))$. 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:

$$
\pi(\mathbf{b}) = \arg\max_a \{Q(\mathbf{b},a) : a \in \mathcal{A}\}\,. \tag{4}
$$

In this way, the stochastic model of the Q-function is effectively transformed into a stochastic policy model, which can be optimised to maximise the reward (Geist and Pietquin, 2011; Gašić et al., 2011; Gašić et al., 2012).

Due to the matrix inversion in Eq. 2, the computational complexity of calculating the Q-function posterior is $O(t^3)$, where t is the number of data points in $\mathbf{B}_t$, and this poses a serious computational problem. The algorithm used here to approximate the Gaussian process is the kernel span sparsification method described in (Engel, 2005). In this case, only a set of representative data points is retained – called the *dictionary* of visited points.

3 BUDS dialogue manager

The Bayesian Update of Dialogue State (BUDS) dialogue manager is a POMDP-based dialogue manager (Thomson and Young, 2010) which factorises the dialogue state into conditionally dependent elements. These elements are arranged into a dynamic Bayesian network, which allows for their marginal probability distributions to be updated during the dialogue. Thus, the belief state of the BUDS dialogue manager consists of the marginal posterior probability distribution over hidden nodes in the Bayesian network. The hidden nodes in the BUDS system consist of the history nodes and the goal nodes for each concept in the dialogue. For instance in a restaurant information domain these include *area*, *food-type*, *address*. The history nodes define possible dialogue histories for a particular concept, eg. *system-informed*, *user-requested*. The goal nodes define possible values for a particular concept, eg. *Chinese*, *Indian*. The role of the policy π is then to map each belief state into a summary action a from the summary action space $\mathcal{A}$. Once a summary action is found it is heuristically mapped into the master action that the system finally takes (Gašić et al., 2012). The master actions are composed of dialogue act type and list of slot value pairs. There are 15 dialogue act types in the BUDS system that facilitate not only simple information providing scenarios but also more complex dialogues where the user can change their mind and ask for alternatives.

To apply GP policy optimisation, a kernel function must be defined on both the belief state space $\mathcal{B}$ and the action space $\mathcal{A}$. The kernel function over the belief state $\mathbf{b}$ is constructed from the sum of individual kernels over the hidden node distributions, such that the kernel function of two corresponding nodes is based on the expected likelihood kernel (Jebara et al., 2004), which is also a simple linear inner product:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}') = \sum_h \langle \mathbf{b}_h, \mathbf{b}'_h \rangle, \qquad (5)$$

where $\mathbf{b}_h$ is the probability distribution encoded in the hth hidden node. This kernel gives the expectation of one belief state distribution under the other.

For history nodes, the kernel is a simple inner product between the corresponding node distributions. 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 represents 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.

For the action space kernel, the δ-kernel is used defined by:

$$k_{\mathcal{A}}(a, a') = \delta_a(a'). \qquad (6)$$

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

3.1 TopTable domain

The TopTable domain consists of restaurants in Cambridge, UK automatically extracted from the TopTable web service (TopTable, 2012). There are about 150 restaurants and each restaurant has 7 attributes – slots. This results in a belief space that consists of 25 concepts where each concept takes from 3 to 150 values and each value has a probability in $[0, 1]$. The summary action space consists of 16 summary actions.

3.2 The agenda-based simulated user

In training and testing a simulated user was used. The agenda-based user simulator (Schatzmann, 2008; Keizer et al., 2010) factorises the user state into an *agenda* and a *goal*. The goal ensures that the user simulator exhibits consistent, goal-directed behaviour. The role of the agenda is to elicit the dialogue acts that are needed for the user simulator to fulfil the goal. In addition, an error model adds confusions to the simulated user input such that it resembles those found in real data (Thomson et al., 2012). The length of the N-best list was set to 10 and the confusion rate was set to 15% during training and testing.[1] This error rate means that 15% of time the true hypothesis is not in the N-best list. Intermediate experimentation showed that these confusion rates are typical of real data.

The reward function was set to give a reward of 20 for successful dialogues, zero otherwise. In addition, 1 is deducted for each dialogue turn to encourage shorter dialogues. The discount factor γ is set to 1 and the dialogue length is limited to 30 turns.

4 Extended domains

Transfer learning is a reinforcement learning technique which address three problems:

- given a target domain, how to select the most appropriate source domain from a set of source domains,
- given a target and a source domain how to find the relationship between them, and
- given a target and a source domain and the relationship between them, how to effectively transfer knowledge between them.

[1]Except of course where the system is explicitly tested on varying noise levels.

Here we assume that we are given a source and a target domain and that the relationship between them is defined by mapping the kernel function. Knowledge transfer is then effected by adapting the source domain policy for use in the target domain. For the latter, two forms of adaptation are investigated: one simply continues to update the set of source data dictionary points with new dictionary points, the second uses the source domain posterior as a prior for the new target domain.

In this case, the source is a basic restaurant domain with slots *name*, *area*, *food-type*, *phone*, *address*, and *postcode*. The extended target domain has an additional *price-range* slot. We are interested primarily in training the policy on the basic domain and testing it on the extended domain. However, since real applications may also require a slot to be *forgotten*, we also investigate the reverse where the policy is trained in the extended domain and tested on the basic domain.

In order to enable the required cross domain portability, a kernel function defining the correlation between belief states from differing domains is needed. Since the extended domain has an extra slot and thus extra hidden nodes, we need to define the correlations between the extra hidden nodes and the hidden nodes in the belief state of the basic domain. This can be performed in various ways, but the simplest approach is to specify which slot from the basic domain is most similar to the new slot in the extended domain and then match their corresponding hidden nodes. In that way the belief state kernel function between two belief states $\mathbf{b}^{\mathsf{B}}$, $\mathbf{b}^{\mathsf{E}}$ for the basic B and the extended E domain becomes:

$$k_{\mathcal{B}}(\mathbf{b}^{\mathsf{B}}, \mathbf{b}^{\mathsf{E}}) = \sum_{h \in \mathsf{B}} \langle \mathbf{b}^{\mathsf{B}}_h, \mathbf{b}^{\mathsf{E}}_h \rangle + \sum_{e \notin \mathsf{B}} \langle \mathbf{b}^{\mathsf{B}}_{l(e)}, \mathbf{b}^{\mathsf{E}}_e \rangle, \quad (7)$$

where h are the hidden nodes in the basic domain, e are the hidden nodes in the extended domain and function $l : \mathsf{E} \to \mathsf{B}$ for each hidden node that does not exist in the basic domain finds its appropriate replacement. In the particular case studied here, the slot *area* is most similar to the new *price-range* slot since they both have a relatively small number of values, about 5. Hence, $l(\textit{price-range}) \to \textit{area}$. If the cardinality of the mapped slots differ, the shorter is padded with zeros though other forms of normalisation are clearly possible.

The (summary) action space for the extended domain has more actions than the basic domain. For example, one action that exists in the extended domain and does not exist in the basic domain is *request(price-range)*. To define the kernel function between these sets of actions, one can specify for each extra action in the extended domain its most similar action in the basic domain:

$$k_{\mathcal{A}}(a^{\mathsf{B}}, a^{\mathsf{E}}) = \begin{cases} \delta_{a^{\mathsf{B}}}(a^{\mathsf{E}}) & a^{\mathsf{E}} \in \mathcal{A}^{\mathsf{B}}, \\ \delta_{a^{\mathsf{B}}}(L(a^{\mathsf{E}})) & a^{\mathsf{E}} \notin \mathcal{A}^{\mathsf{B}}, \end{cases} \quad (8)$$

where function $L : \mathcal{A}^{\mathsf{E}} \to \mathcal{A}^{\mathsf{B}}$ for each action that does not exist in the basic domain finds its replacement action.

Functions L and l are here defined manually. However, a simple but effective heuristic would be to find for each new slot in the extended domain, a slot in the basic domain with similar cardinality.

Porting in the reverse direction from the extended to the basic domain is easier since one can simply disregard the extra hidden nodes and actions in the kernel calculation.

To experimentally examine the extent to which this method supports cross domain portability, we trained policies for both domains until convergence, using 10^5 dialogues on the simulated user. We then cross tested them on the mismatching domains at varying user input error rates. The results are given in Fig. 1.

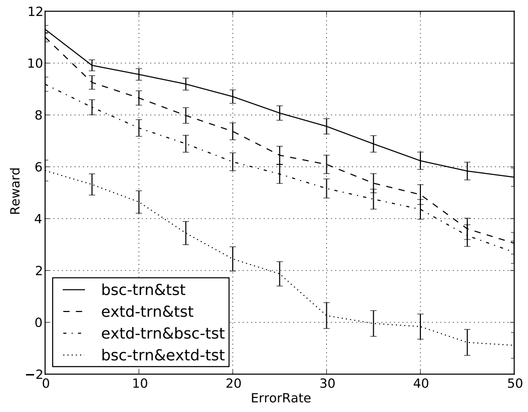

Figure 1: Cross testing policies trained on different domains. bsc refers to the basic domain, extd is the extended domain, trn is training and tst is testing.

From the results it can be seen that the policy trained for the basic domain has a better performance than the policy trained on the extended domain, when tested on the matching domain (compare bsc-trn&tst with extd-trn&tst). The extended domain has more slots so it is more difficult for the system to fulfil the user request, especially in noisy conditions. Secondly, the performance of the policy trained on the extended domain and tested on the basic domain is close to optimal (compare bsc-trn&tst with extd-trn&bsc-tst). However, the policy trained on the basic domain and tested on the extended domain has much worse performance (compare bsc-trn&extd-tst with extd-trn&tst). It is hard for the policy to adequately extrapolate from the basic to the extended domain. This difference in performance, however, motivates the need for adaptation and this is investigated in the next section.

5 Adaptation

Adaptation of a policy trained on one domain to another can be performed in several ways. Here we examine two adaptation strategies similar to the method described in (Taylor et al., 2007), where every action-value for each state in the target domain is initialised with learned source domain values.

The first strategy is to take the policy trained in the source domain and simply continue training it in the target domain until convergence. In Gaussian process reinforcement learning, this means that we assume a zero-mean prior on the Gaussian process for the Q-function and let the dictionary of visited points $\mathbf{B}_t$ from Eq. 2 consist of both points visited in the source domain and the extended target domain, making sure that the Gram matrix $\mathbf{K}_t$ uses extended domain kernel function where necessary. However, the estimate of the variance decreases with the number of visited points (see Eq. 2). The danger therefore when performing adaptation in this way is that the estimate of variances obtained in the source domain will be very small since the policy has already been trained until convergence with a large number of dialogues. As a consequence, the rate of exploration defined by sampling in Eq. 4 will be reduced and thus lead to the subsequent optimisation in the new target domain falling prematurely into a local optimum.

As an alternative, we propose another adaptation strategy. The estimate of the posterior of the mean for the Q-function, $\overline{Q}$ in Eq. 2, from the policy trained on the basic domain can be taken to be the prior of the mean when the policy is trained on the extended domain as in Eq. 3. More precisely, if $\overline{Q}_{\text{bsc}}$ is the posterior mean of the policy trained on the basic domain then $m_{\text{extd}} = \overline{Q}_{\text{bsc}}$. In this case it is also important to make sure that the kernel function used to calculate $\overline{Q}_{\text{bsc}}$ is redefined for the extended domain where necessary. The prior on the variance is the original kernel function renormalised:

$$k((\mathbf{b}, a), (\mathbf{b}', a')) \leftarrow \frac{k((\mathbf{b},a),(\mathbf{b}',a'))}{\sqrt{k((\mathbf{b},a),(\mathbf{b},a))k((\mathbf{b}',a'),(\mathbf{b}',a'))}}. \quad (9)$$

Given that the estimate of the mean provides reasonable performance, it is not necessary to place a flat prior on the variance of the Q-function and therefore the kernel is normalised as in Eq. 9.

When comparing adaptation strategies, we are interested in two aspects of performance. The first is the performance of the policy during training. The second is how quickly the policy reaches the optimal performance. For that reason we adopt the following evaluation scheme. After every 100 adaptation dialogues we test the partially optimised policy with 1000 simulated dialogues, different to the ones used in adaptation. These 1000 dialogues are the same for every test point on the graph. The results are given in Fig. 2.

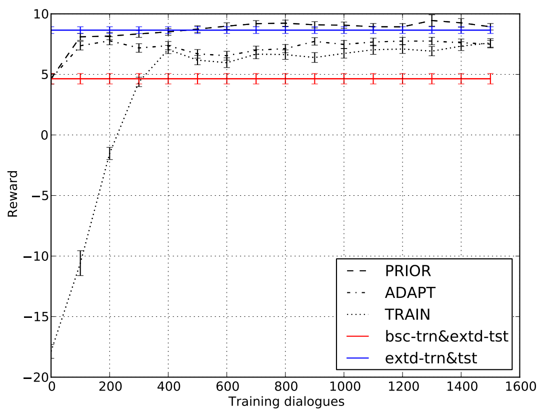

Figure 2: Different adaptation strategies

The lower horizontal line represents the performance of the policy trained on the basic source domain and tested on the extended target domain. This is the baseline. The upper horizontal line represents the policy trained until convergence on the extended domain and also tested on the extended domain. This provides the gold standard. The adaptation strategy that takes both the mean and variance of the policy trained on the basic domain and retrains the policy on the extended domain is denoted as ADAPT in Fig. 2. The adaptation strategy that uses the posterior mean of the policy trained on the source domain as the prior mean for adaptation is denoted as PRIOR in Fig. 2. Finally, for comparison purposes we show the performance of the policy that is trained from scratch on the extended domain. This is denoted as TRAIN on the graph. It can be seen that both adaptation strategies significantly reduce the number of training dialogues and, more importantly, maintain the level of performance during adaptation. The adaptation strategy that places the prior on the mean has slightly worse performance in the beginning but provides the best performance after 1500 dialogues. As already noted, this could be due to overly confident variances in the ADAPT case leading to a local optimum.

6 Human experiments

In order to adapt and evaluate policies with humans, we used crowd-sourcing via the Amazon Mechanical Turk service in a set-up similar to (Jurčíček et al., 2011a; Gašić et al., 2013). The BUDS dialogue manager was incorporated in a live telephone-based spoken dialogue system. The Mechanical Turk users were assigned specific tasks in the extended TopTable domain. They were asked to find restaurants that have particular features as defined by the given task. To elicit more complex dialogues, the users were sometimes asked to find more than one restaurant, and in cases where such a restaurant did not exist they were required to seek an alternative, for example find a Chinese restaurant instead of a Vietnamese one. After each dialogue the users filled in a feedback form indicating whether they judged the dialogue to be successful or not. Based on that binary rating, the subjective success was calculated as well as the average reward. An objective rating can also be obtained by comparing the system outputs with the predefined task.

During policy adaptation, at the end of each call, users were asked to press 1 if they were satisfied (i.e. believed that they had been successful in fulfilling the assigned task) and 0 otherwise. The objective success was also calculated. The dialogue was then only used for adaptation if the user rating agreed with the objective measure of success as in (Gašić et al., 2013). The performance based on user ratings during adaptation for both adaptation strategies is given in Table 1.

Table 1: Policy performance during adaptation

	#Diags	Reward	Success (%)
ADAPT	251	11.7 ± 0.5	92.0 ± 1.7
PRIOR	329	12.1 ± 0.4	96.7 ± 1.0

We then evaluated four policies with real users: the policy trained on the basic domain, the policy trained on the extended domain and the policy adapted to the extended domain using the prior and the policy adapted to the extended domain via interaction with real users using retraining. The results are given in Table 2.

Table 2: Human evaluation of four systems in the extended domain: trained in the basic domain, trained in the extended domain, trained in the basic and adapted in the extended domain using both ADAPT and PRIOR methods.

Training	#Diags	Reward	Success(%)
Basic	246	11.0 ± 0.5	91.9 ± 1.7
Extended	250	12.1 ± 0.4	94.4 ± 1.5
ADAPT	268	12.6 ± 0.4	94.4 ± 1.4
PRIOR	252	12.4 ± 0.4	95.6 ± 1.3

The results show two important features of these adaptation strategies. The first is that it is possible to adapt the policy from one domain to another with a small number of dialogues. Both adaptation techniques achieve results statistically indistinguishable from the matched case where the policy was trained directly in the extended domain. The second important feature is that both adaptation strategies guarantee a minimum level of performance during training, which is better than the performance of the basic policy tested on the extended domain. This is particularly important when training with real users so that they are not exposed to poor performance at any time during training.

7 Application to fast learning

The above results show that transfer learning through policy adaptation can be relatively fast. Since complex domains can be decomposed into a series of domains with gradually increasing complexity, an alternative to training a system to convergence starting from an uninformative prior is to train a system in stages iteratively adapting to successively more complex domains (Taylor and Stone, 2009).

We explored this idea by training the extended system in three stages. The first has only one slot that the user can specify: *food-type* and additional slots *phone*, *address* and *postcode* that can be requested (initial in Fig. 3). The second has an additional *area* slot (intermediate in Fig. 3) and the final domain has a the *price-range* slot added (final on the graph).

A policy for each of these domains was trained until convergence and the average rewards of these policies are the horizontal lines on Fig. 3. In addition, the following adaptation schedule was implemented. An initial policy was trained from scratch for the one-slot initial system using only 1500 dialogues. The resulting policy was then retrained for the intermediate two-slot system using again just 1500 dialogues. Finally, the required three-slot system was trained using 1500 dialogues. At each stage the policy was tested every 100 training dialogues, and the resulting performances are shown by the three graphs initial-train, intermediate-adapt and final-adapt in Fig. 3. The policies were tested on the domains they are trained on or adapted to.

It can be seen that after just 500 dialogues of the third stage (i.e. after just 3500 dialogues in total) the policy reaches optimal performance. It has been shown previously that Gaussian process reinforcement learning for this task normally takes 10^4 dialogues (Gašić et al., 2012) so this schedule halves the number of dialogues needed for training. Also it is important to note that when training from scratch the average reward is less than 5 for 300 dialogues (see TRAIN in Fig. 2), in this case that only happens for about 100 dialogues (see initial-train in Fig. 3).

8 Conclusions

This paper has investigated the problem of extending a dialogue system to handle new previously unseen concepts (i.e. slots) using adaptation based transfer learning. It has been shown that a GP kernel can be mapped to establish a relationship between a basic and an extended domain and that GP-based adaptation can restore a system to optimal performance within 200 to 300 adaptation dialogues. A major advantage of this technique is that it allows a minimum level of performance to be guaranteed and hence guards against subjecting the user to poor performance during the early stages of adaptation.

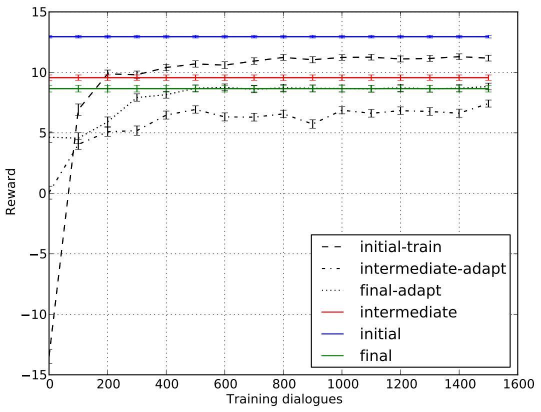

Figure 3: Application of transfer learning to fast training. The target is to achieve the performance of the fully trained 3 slot system as shown by the lower horizontal line final. This is achieved in three stages, with the target being achieved part way through the 3rd stage using just 3500 dialogues in total.

Two methods of adaptation have been studied – one based on augmenting the training points from the source domain with new points from the target domain, and a second which treats the source policy as a prior for the target policy. Results using the prior method were consistently better. In a further experiment, it was also shown that starting with a simple system and successively extending and adapting it slot by slot, can achieve optimal performance faster than one trained directly from scratch.

These results suggest that it should be feasible to construct dialogue systems which can dynamically update and extend their domains of discourse automatically during direct conversations with users. However, further investigation of methods for learning the relationship between the new and the old domains is needed. Also, the scalability of these results to large-scale domain expansion remains a topic for future work.

Acknowledgments

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

References

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

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

M Gales and S Young. 2007. The application of hidden Markov models in speech recognition. *Found. Trends Signal Process.*, 1:195–304.

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

M Gašić, F Jurčíček, B Thomson, K Yu, and S Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Proceedings of ASRU*.

M Gašić, M Henderson, B Thomson, P Tsiakoulis, and S Young. 2012. Policy optimisation of POMDP-based dialogue systems without state space compression. In *Proceedings of SLT*.

M Gašić, C. Breslin, M. Henderson, Szummer M., B Thomson, P. Tsiakoulis, and S Young. 2013. On-line policy optimisation of Bayesian Dialogue Systems by human interaction. In *Proceedings of ICASSP*.

M Geist and O Pietquin. 2011. Managing Uncertainty within the KTD Framework. In *Proceedings of the Workshop on Active Learning and Experimental Design*, Sardinia (Italy).

K Georgila and O Lemon. 2004. Adaptive multimodal dialogue management based on the information state update approach. In *W3C Workshop on Multimodal Interaction*.

S Janarthanam and O Lemon. 2010. Adaptive Referring Expression Generation in Spoken Dialogue Systems: Evaluation with Real Users. In *Proceedings of SIGDIAL*.

T Jebara, R Kondor, and A Howard. 2004. Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844, December.

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

F Jurčíček, B Thomson, and S Young. 2011b. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs. *ACM Transactions on Speech and Language Processing*.

S Keizer, M Gašić, F Jurčíček, F Mairesse, B Thomson, K Yu, and S Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDIAL*.

DJ Litman and S Pan. 1999. Empirically evaluating an adaptable spoken dialogue system. In *Proceedings of the seventh international conference on User modelling*.

DJ Litman and S Pan. 2002. Designing and evaluating an adaptive spoken dialogue system. *User Modelling and User-Adapted Interaction*, 12:111–137.

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

N Roy, J Pineau, and S Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proceedings of ACL*.

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

ME Taylor and P Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December.

ME Taylor, P Stone, and Y Liu. 2007. Transfer learning via inter-task mappings for temporal difference learning. *J. Mach. Learn. Res.*, 8:2125–2167, December.

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

B Thomson, M Gašić, M Henderson, P Tsiakoulis, and S Young. 2012. N-Best error simulation for training spoken dialogue systems. In *Proceedings of SLT*.

TopTable. 2012. TopTable. `https://www.toptable.com`.

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

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

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