Natural Actor and Belief Critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs

FILIP JURČÍČEK, BLAISE THOMSON and STEVE YOUNG
Cambridge University

This paper presents a novel algorithm for learning parameters in statistical dialogue systems which are modelled as Partially Observable Markov Decision Processes (POMDPs). The three main components of a POMDP dialogue manager are a dialogue model representing dialogue state information; a policy which selects the system's responses based on the inferred state; and a reward function which specifies the desired behaviour of the system. Ideally both the model parameters and the policy would be designed to maximise the cumulative reward. However, whilst there are many techniques available for learning the optimal policy, no good ways of learning the optimal model parameters that scale to real-world dialogue systems have been found yet.

The presented algorithm, called the Natural Actor and Belief Critic (NABC), is a policy gradient method which offers a solution to this problem. Based on observed rewards, the algorithm estimates the natural gradient of the expected cumulative reward. The resulting gradient is then used to adapt both the prior distribution of the dialogue model parameters and the policy parameters. In addition, the paper presents a variant of the NABC algorithm, called the Natural Belief Critic (NBC), which assumes that the policy is fixed and only the model parameters need to be estimated. The algorithms are evaluated on a spoken dialogue system in the tourist information domain. The experiments show that model parameters estimated to maximise the expected cumulative reward result in significantly improved performance compared to the baseline handcrafted model parameters. The algorithms are also compared to optimization techniques using plain gradients and state-of-the-art random search algorithms. In all cases, the algorithms based on the natural gradient work significantly better.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms: Theory, Algorithms, Experimentation

Additional Key Words and Phrases: Spoken dialogue systems, POMDP, Reinforcement learning

Submitted for the Special Issue of Speech and Language Processing on Machine Learning for Robust and Adaptive Spoken Dialogue Systems

Author's address: F. Jurčíček, B. Thomson, S. Young, Engineering Department, Cambridge University, Trumpington Street, Cambridge CB2 1PZ, UK; email:{fj228, brmt2, sjy}@eng.cam.ac.uk

1. INTRODUCTION

Despite recent developments in gesture-driven visual interfaces, users still have to learn how to use a new interface before they are be able to communicate effectively with the machine. As applications continue to increase in complexity, the limitations of gesture- and menu-driven commands will become more apparent. The learning curve will grow and the constraints caused by the lack of expressiveness of the modality will begin to frustrate users. Speech on the other hand is the primary and the most natural means of communication. It requires no learning and it has significant expressive power. As such, it is likely to become an increasingly significant component of future generation man-machine interfaces.

Current speech enabled applications often automate communication with database systems, especially when a user can only communicate with the application over a phone. Examples of such spoken dialogue systems include: remote access to information and booking systems [Raux et al. 2005], and voice diallers [Williams 2008a]. Whilst these systems are still relatively crude, they exhibit both the potential advantages of speech and the problems which result from imperfect speech recognition.

A typical dialogue system consists of a speech understanding component, a dialogue manager, and a speech generation component. Speech understanding usually consists of a speech recogniser and a semantic parser, and speech generation requires a natural language generator and a speech synthesiser. The main challenge in spoken dialogue systems is to robustly handle communication errors produced by the input/output components of the system, especially those resulting from speech recognition and understanding errors.

In recent years, Partially Observable Markov Decision Processes (POMDPs) have been proposed as a principled way of modelling the inherent uncertainty in spoken dialogue systems [Young et al. 2010]. When interacting with the system, the user's speech is converted to words by the recogniser and passed to the semantic parser which outputs an N-best list of dialogue acts. For example, if a user utters "I want an Indian restaurant in the cheap price range", the corresponding dialogue act would be "inform(food=Indian, type=restaurant, pricerange=cheap)". However, if the understanding component was uncertain about what the user said, it might also output "inform(food=Italian, type=restaurant)", and other variants. This N-best list of dialogue acts is then used by the dialogue manager to update the dialogue state. Based on this updated state, a dialogue policy, and the associated application database, a system action is produced, again in the form of a dialogue act. The system action is then passed to the natural language generator which converts it to text and then via the synthesiser to speech. Readers interested in POMDPs and reinforcement learning in general may refer to [Kaelbling et al. 1998; Sutton and Barto 1998] for more details.

A POMDP dialogue manager includes three main parts: a dialogue model representing state information such as the user's goal, the user's dialogue act and the dialogue history; a policy which selects the system's responses based on the inferred dialogue state; and a reward function which specifies the desired behaviour of the system. In a POMDP system, the dialogue model provides a compact representation for the distribution of the unobserved dialogue state called the *belief state*

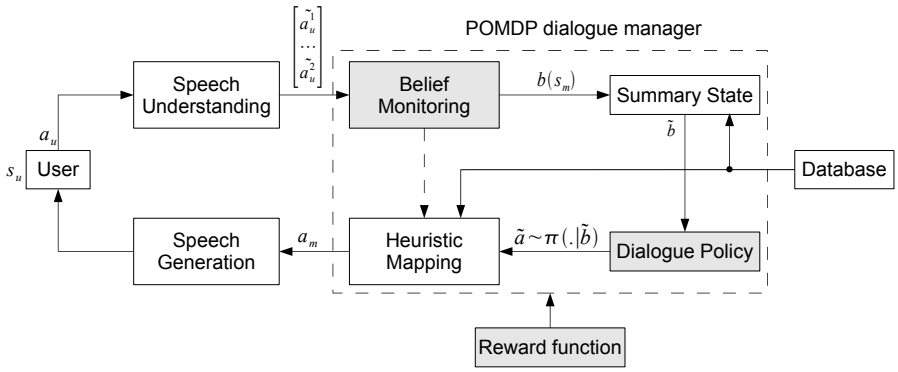

Fig. 1. Structure of a POMDP dialogue system: a_u is a user act produced given the user state s_u, $\tilde{a}_u^i$ is an estimate of the users' dialogue acts, $b(s_m)$ is the belief state, $\tilde{b}$ is a summary representation of the belief state, $\tilde{a}$ is a summary action which is later expanded into full dialogue act, a_m, by a heuristic mapping based on the full belief state and the database.

and it is updated every turn based on the observed user inputs in a process called *belief monitoring*. Exact belief monitoring of the full dialogue state is intractable for all but the simplest systems. One way to address this issue is to represent the state in the compact and approximate form of a dynamic Bayesian Network (BN), where nodes represents attributes[1] in the application ontology. Then by exploiting the conditional independence of the network nodes, a tractable system can be built [Thomson and Young 2010]. In this case, the parameters of the model describe the conditional distributions of the nodes in the network.

The policy selects the dialogue system's responses (actions) based on the belief state at each turn, and it is typically trained using reinforcement learning with the objective of maximising the expected cumulative reward. The use of reinforcement learning algorithms for POMDP systems usually relies on the observation that a POMDP system can be transformed into a continuous state Markov decision process (MDP) and that the policy optimization problem can then be solved for this newly defined MDP with the guarantee that the solution also optimises the original POMDP [Kaelbling et al. 1998]. However, without approximations this is not tractable for any real-world dialogue system.

Significant reduction in complexity can be achieved if notions of a summary space and summary actions are introduced. The basic idea is that a successful policy does not need access to all of the information in the belief state and the database, and that summary actions produced by the policy can be mapped back into full actions through a heuristic mapping [Williams and Young 2005]. A typical structure of a POMDP dialogue manager embodying these ideas is depicted in Figure 1.

While there are many efficient techniques for learning the policy parameters [Sutton and Barto 1998; Peters et al. 2005; Engel et al. 2005; Geist et al. 2009], no good

[1]Attributes are constraint values such as type, pricerange, food, etc. In simple spoken dialogue systems which do not make use of an ontology, they are often referred to as slots. Here the terms slot and attribute should be regarded as synonyms.

ways of learning the model parameters which scale to real-world dialogue systems have been established yet. Hence, in virtually all current systems, the dialogue model parameters are hand-crafted by a system designer [Roy et al. 2000; Zhang et al. 2001; Bui et al. 2009; Thomson and Young 2010; Young et al. 2010]. Ideally, one would like to estimate the parameters from the interactions with the user and some attempts have been made in this direction. For example, Georgila et al. [2005] and Kim et al. [2008] used maximum likelihood estimates from an automatically annotated corpus of real dialogues. The method is problematic, however, since the quality of the estimated model depends on the quality of the automatic annotation which itself relies on having a good dialogue model. Williams [2008b] obtained maximum likelihood estimates from a corpus with manually annotated dialogue states. Although this approach can provide good estimates of the dialogue model, it is very laborious and in practice only a small number of dialogues can be obtained. Doshi and Roy [2007] presented a dialogue model which was trained by Viterbi learning and using a reinforcement signal based on the successful completion of each dialogue. In this method, the initial model parameters have to be handcrafted with some reasonable quality to be able to improve the initial parameter parameters. Syed and Williams [2008] showed how to use Expectation-Maximization (EM) to learn parameters of an observation model. Please note that the dialogue model can be factored into a transition model between states and an observation model. In this work, no attempts were made to learn the transition model. By assuming that the user goal remains constant, the complexity of the problem was significantly reduced and a tractable method was achieved. Thomson [2010] used Expectation-Propagation (EP) to infer hidden state information together with the model parameters. The algorithm treated the task of learning the model parameters in an unsupervised manner. However, it is not clear to what extent likelihood maximisation over a dialogue corpus correlates with the expected cumulative reward of the dialogue system.

This paper presents a novel reinforcement algorithm called Natural Actor and Belief Critic (NABC) for jointly learning the parameters of a dialogue model and a policy which maximises the expected cumulative reward. The method is presented and evaluated in the context of the BUDS POMDP dialogue manager which uses a dynamic Bayesian Network to represent the dialogue state. However, the method is sufficiently general that it could be used to optimise virtually any parameterised dialogue model. Furthermore, unlike most of the maximum likelihood methods used so far, the NABC algorithm does not require that the user goal remains constant or that the model be generative.

The second algorithm presented in this paper is the Natural Belief Critic algorithm which is a variant of the original NABC algorithm. The NBC algorithm assumes that the policy is fixed and only the model parameters need to be learnt [Jurčíček et al. 2010]. For example, the NBC algorithm can be used to optimise the model parameters in a dialogue system using a hand-crafted policy.

The rest of the paper is structured as follows. Section 2 briefly describes the BUDS dialogue manager and the method it uses for policy representation [Thomson and Young 2010]. Section 3 then describes policy gradient methods and a specific form called the episodic Natural Actor Critic (eNAC) algorithm which is used to

optimise the BUDS policy. In Section 4.1, the proposed Natural Actor and Belief Critic algorithm is presented as a generalisation of the eNAC algorithm which is then followed by a description of the Natural Belief Critic algorithm in Section 4.2. To be able to use either the NABC or NBC algorithms, a suitable prior for the dialogue model parameters must be defined. This mostly depends on the model used in belief monitoring. The prior used for the BUDS model parameters is described in Section 4.3. Both algorithms are evaluated on a system designed for the tourist information domain in Section 5. The results and alternative methods are discussed in Section 6. Finally, Section 7 presents conclusions.

2. BUDS DIALOGUE MANAGER

In a POMDP dialogue system, the true dialogue state s_t is unknown. Therefore, the policy selects an action a_t at time t based on the distribution over all states called the belief state, b_t. The estimate of the belief state depends on past observations and actions, which are referred to as the observed history $h_t = \{a_0, o_1, \ldots, a_{t-1}, o_t\}$. If the system is Markovian then the belief state b_t depends only on the previous belief state b_{t-1}, the current observation o_t and the last system action a_{t-1}. Using Bayes theorem, the belief state b_t can be computed as follows:

$$\begin{aligned} b_t &= b(s_t|h_t;\tau) \\ &= k \cdot p(o_t|s_t;\tau) \sum_{s_{t-1}} p(s_t|a_{t-1}, s_{t-1};\tau) b(s_{t-1}|h_{t-1};\tau) \end{aligned} \tag{1}$$

where the transition probability function $p(s_t|a_{t-1}, s_{t-1};\tau)$ and the observation probability $p(o_t|s_t;\tau)$ represent the dialogue model which is parameterised by τ and k is a normalisation constant [Kaelbling et al. 1998].

2.1 The observations and system actions

In the BUDS dialogue manager, the observations and the system actions are implemented as dialogue acts. A dialogue act conveys the user or system intention. In BUDS, the dialogue acts take the form $actt(a_1 = v_1, a_2 = v_2, \ldots)$ where $actt$ denotes the type of dialogue act and the arguments are the slot-value pairs where slots refer to nodes in the user goal described in the next section. In some cases, the value can be omitted, for example, where the intention is to query the value of a slot e.g. "request(food)". A full description of the dialogue act set used by the BUDS system is given in [Young 2007].

When there is uncertainty in the speech understanding process, the input to the dialogue manager is a list of alternative dialogue acts. For example, the utterance "I want an Indian restaurant in the cheap price range" spoken in a noisy background might yield

```
inform(food=Indian, type=restaurant, pricerange=cheap) {0.6}
inform(food=Italian, type=restaurant) {0.4}
```

where the numbers in the brackets represent the probabilities of each dialogue act.

2.2 The dialogue model

A naive implementation of (1) is not tractable since there are billions of states in a real-world spoken dialogue system[2]. Thus, the BUDS dialogue manager uses a Bayesian Network (BN) to represent the state of the POMDP system, where the network nodes represent the slots in the system [Thomson and Young 2010]. Provided that each slot or network node has only a few dependencies, tractable systems can be built and belief estimates maintained with acceptable accuracy using approximate inference [Bishop 2006].

The BUDS dialogue state is factored into three components: the user goal g, the user action u and the dialogue history d. In addition, the goal and the history are further factored into sub-goals g_k and sub-histories d_k according to a set of slots, $k \in \mathcal{K}$, in the system. For example, in a tourist information system typical sub-goals might be the type of venue required ("type") or the type of food ("food"). The sub-history nodes allow the system designer to store information about the history in order to make coherent decisions. In the BUDS system, the sub-history nodes have three values: "nothing-said", "system-informed", and "user-requested". The user action u is the estimate of the true dialogue act from the observation o. Apart from the task specific sub-goals, the dialogue model can include additional nodes. For example, in a tourist information system the "method" node helps the dialogue manager to decide whether a user is searching for a venue by attribute or by name. Also, the dialogue model includes a "discourse" node to model discourse actions and to control the dialogue flow. For example, the discourse node is used to infer whether a user wants to repeat the last system action, restart the dialogue, or end the dialogue. The reader interested in the design details of the dialogue model may refer to [Thomson 2010].

The topology of the Bayesian network depends on various conditional independence assumptions. The user act depends on all sub-goals and the last system action. The sub-goals depend on their previous value, the last system action and may depend on any combination of the other current sub-goals. The sub-history depends on the previous sub-history and the current user act. Figure 2 shows the resulting network for two time-slices of a two-slot system based on this idea. The network also includes observed system actions a and observations o.

The BUDS system's inference algorithm uses grouped loopy-belief propagation [Thomson and Young 2010]. It maintains marginal belief estimates only for slot values which were mentioned by the user. Those slot values which were not mentioned are assumed to belong to one group with a uniform probability distribution within this group. Then, the belief update involves only the recomputation of probabilities of the mentioned slot values and one probability of the grouped values. This grouped loopy-belief propagation is particularly efficient in situations where the number of potential slot values is large but the user typically only ever mentions a few of them.

The BN model parameters τ comprise the set of conditional probabilities of the node values. For example, the "food" sub-goal values are described by the prob-

[2]Note that if a dialogue system has 10 slots and each slot has 10 different values then there are 10^{10} distinct states.

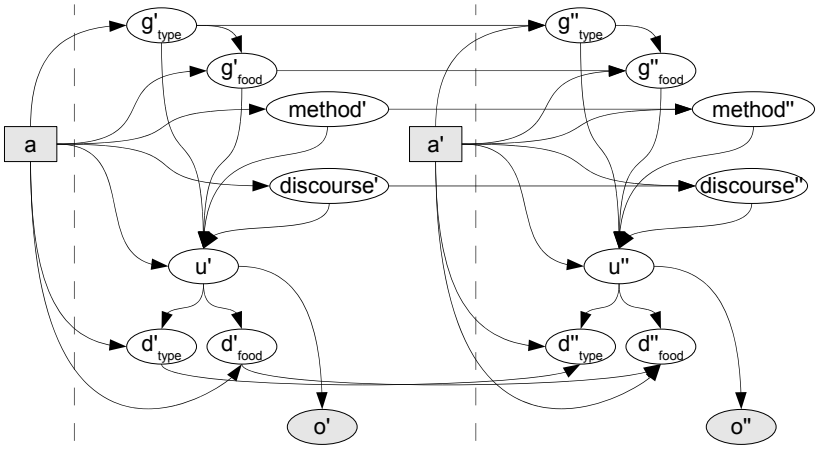

Fig. 2. An example factorisation for the Bayesian network representing part of a tourist information dialogue system. The white nodes are unobserved variables while the grey nodes denote observed variables. The squares represents the system actions decided by a dialogue policy.

ability $p(g'_{food}|g_{food}, g'_{type}, a; \tau_{food})$ parameterised by τ_{food}. To reduce the number of parameters specifying the distributions in the sub-goals, some parameters are tied together on the assumption that the probability of change in the sub-goals is constant given the last system action and the parent sub-goal. For example, the probability of change from "Chinese" to "Indian" in the sub-goal "food" is assumed to be equal to the probability of change from "Chinese" to "Italian". As a result, instead of having N^2 parameters, where N is the number of values in the sub-goal, only one probability of change $\tau_{food,change}$ is needed. The probability of the sub-goal staying the same is equal to $1 - \tau_{food,change}$. The sub-history, "discourse" and "method" nodes do not use parameter tying as the ability to set different values for the probability of change is beneficial in this case.

2.3 The Policy

The BUDS dialogue manager uses a stochastic policy $\pi(a|b;\theta)$ which gives the probability of taking action a given belief state b and policy parameters θ. When used in the dialogue manager, the policy distribution is sampled to yield the required action at each turn. To reduce complexity, for every action a, the belief state is mapped into a vector of features, $\Phi_a(b)$ and the policy is then approximated by a softmax function:

$$\pi(a_t|b(\cdot|h_t;\tau);\theta) \approx \frac{e^{\theta^T \cdot \Phi_{a_t}(b(\cdot|h_t;\tau))}}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b(\cdot|h_t;\tau))}}. \quad (2)$$

To estimate the policy parameters, BUDS uses the episodic Natural Actor Critic (eNAC) algorithm [Peters et al. 2005] (see Section 3).

A further reduction in complexity can be achieved by utilising summary actions [Thomson and Young 2010]. The full set of actions is not needed as some actions will be always sub-optimal. For example, if the dialogue manager confirms the value of some sub-goal then it should always confirm the most likely value. As a

result, the full set of actions is not needed. The BUDS dialogue manager uses the following slot-level summary actions: "request", "confirm" the most likely value, and "select" between the two most likely values. In addition, it supports global actions such as "inform", "repeat", and "bye". The mapping of the summary actions into full dialogue acts is performed by a hand-crafted function based on the information in the belief state and the database. For example, the summary action "confirm food" is mapped to "confirm(food=Chinese)" provided that the value "Chinese" is the most probable value in the sub-goal "food". The database is used when the system informs about a particular venue. For example, the inform summary action "inform name pricerange" results in "inform(name="Graffiti Restaurant",pricerange=cheap)", where the information about the price range of the restaurant "Graffiti Restaurant" is retrieved from the database.

There are a variety of possible forms for the Φ function. Ideally, it should perform some form of tiling [Sutton and Barto 1998]. However, this is not always necessary and simpler forms of feature extraction can be considered. The BUDS dialogue manager uses a slot-level grid-based approximation [Thomson 2010]. First, it is assumed that the slot-level summary actions depend mainly on the probability distribution in a particular slot. Then, the probability distribution is mapped into a binary vector which indicate nearness of the probabilities of the two most likely values to one of the predefined grid points. For example, the probabilities of the two most likely slot values can be formed into a tuple, e.g. $(0.8, 0.1)$. Then the closest grid point is identified from the following set $\{(1.0, 0.0), (0.8, 0.0), (0.8, 0.2), (0.6, 0.0), (0.6, 0.2), (0.6, 0.4), (0.0, 0.0)\}$ according to the L_2 distance. Finally, the index of the closest grid point is mapped into a 7-dimensional binary vector with 1 assigned to the corresponding position in the vector and 0 everywhere else. In addition, there are also features for actions such as "inform about the matching venue" or "inform about the requested slots". For example, the features include the number of accepted slots in the Bayesian Network and the number of matching venues in the database given the accepted slots. From the point of view of the dialogue manager, an accepted slot represents a piece of information which can be trusted and used to query the database. In BUDS, a slot is regarded as being accepted if its most likely value has a probability higher than some threshold, in our case 0.8.

BUDS also supports hand-crafted policies which are designed by an expert. These policies deterministically choose which action to take given the belief state and they have the form of if/then statements written in the source code of the dialogue manager. An example of a simple but reasonable hand-crafted policy is as follows:

(1) request values for slots in which the most likely value has probability lower than 0.3,

(2) confirm the most likely values in slots where the most likely value has probability between 0.3 and 0.9,

(3) select between the two most likely values if the sum of the probabilities for these values is more than 0.8,

(4) accept every slot which has a probability of the most likely value higher than 0.9,

(5) inform about a venue if there is only one matching venue (matching is based on the accepted slots),

(6) inform about slots requested by a user if the probability of the "user-requested" value of the respective sub-history nodes is more than 0.5.

3. POLICY GRADIENTS

The objective of reinforcement learning is to find a policy π which maximises the expected cumulative reward $J(\theta)$:

$$J(\theta) = E\left[\sum_{t=0}^{T-1} r(b_t, a_t) \mid \pi_\theta\right],$$

where $r(b_t, a_t)$ is the immediate reward when taking action a_t in belief state b_t and T is the number of turns in a dialogue[3].

Learning the policy parameters θ can be achieved by a gradient ascent which iteratively adds a multiple of the gradient to the parameters being estimated [Sutton et al. 2000]. Using "the log likelihood-ratio trick" [Williams 1992] and Monte Carlo sampling, the gradient can be estimated as follows:

$$\nabla J(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta) Q^\pi(b_t^n, a_t^n) \tag{3}$$

where the sampled dialogues are numbered $n = 1 \dots N$, the n-th dialogue has a length of T_n turns, and $Q^\pi(b, a)$ is the value of a belief-state/action pair given the policy: $Q^\pi(b, a) = E[\sum_{k=0}^{T-1} r(b_{t+k}, a_{t+k}) \mid b_t = b, a_t = a, \pi_\theta]$. Note that $Q^\pi(b, a)$ can be approximated by a function compatible with the policy without affecting the unbiasedness of the gradient [Sutton et al. 2000; Konda and Tsitsiklis 2000].

To obtain a closed form solution for (3), the policy π must be differentiable w.r.t. θ. Conveniently, the softmax function (2) uses the policy parameters θ in a scalar product with the feature vector $\Phi_a(b)$. Thus, it is easy to derive an analytic form for the gradient ∇J. A detailed derivation of $\nabla_\theta \log \pi(a_t|b(\cdot|h_t;\tau);\theta)$ is given in Appendix A.

Although (3) can provide an estimate for the "plain" gradient, it has been shown that the natural gradient $\widetilde{\nabla} J(\theta) = F_\theta^{-1} \nabla J(\theta)$ is more effective for optimisation of statistical models where F_θ is the Fisher Information Matrix [Amari 1998]. Based on this idea, Peters et al. [2005] developed a family of Natural Actor Critic algorithms which estimate the *natural gradient* of the expected cumulative reward. The appealing part of these algorithms is that in practice the Fisher Information Matrix does not need to be explicitly computed. In this work, the episodic version the Natural Actor Critic algorithm is used to train the BUDS's stochastic policy. To obtain the natural gradient, w, of $J(\theta)$, the episodic Natural Actor Critic algorithm

[3]In the BUDS dialogue system, only episodic tasks with finite number of turns are considered. In the case of non-episodic tasks, one can use an expected discounted reward $J(\theta) = E[\sum_{t=0}^{\infty} \gamma^t r(b_t, a_t) \mid \pi_\theta]$ where γ is a discounting factor between 0 and 1.

(eNAC) uses a least square method to solve the following set of equations:

$$r_n = \left[\sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n | b(\cdot|h_t^n;\tau);\theta)^T \right] \cdot w + C \quad \forall n \in \{1,\ldots,N\}, \tag{4}$$

where r_n is the reward observed at the end of a dialogue n. As the episodic Natural Actor Critic algorithm assumes that there is a single initial state of the system, the constant C from (4) can be interpreted as the expected cumulative reward of the initial state. Once w has been found, the policy parameters can be iteratively improved by $\theta' \leftarrow \theta + \beta w$, where β is a step size.

When the eNAC algorithm is compared to the gradient approach outlined in [Sutton et al. 2000], the eNAC algorithm is significantly more efficient suggesting that the use of the natural gradient is critical [Konda and Tsitsiklis 2000]. The question therefore arises whether this type of policy gradient method can be generalised to optimise not just the policy but the parameters of the dialogue model as well.

4. LEARNING THE DIALOGUE MODEL PARAMETERS

4.1 Natural Actor and Belief Critic algorithm

The difficulty with using policy gradient methods for learning the parameters of the dialogue model 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 w.r.t. 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 w.r.t. the parameters α. This leads to a generalisation of the eNAC algorithm called the Natural Actor and Belief Critic (NABC) algorithm.

The goal of NABC is to learn the parameters α of the prior distribution for the model parameters together with the policy parameters θ while maximising the expected cumulative reward. To derive the algorithm, a similar approach to that taken for policy gradients will be used [Wierstra et al. 2007; Peters and Schaal 2008a]. Let the model parameters τ be sampled at the beginning of each dialogue, then τ becomes part of the observed history: $h_t = \{\tau, a_0, o_1, \ldots, a_{t-1}, o_t\}$. Let H_t be the trajectory[4] of the system which consists of the observed history together with the unobserved states: $H_t = \{\tau, s_0, a_0, o_1, s_1, \ldots, a_{t-1}, o_t, s_t\}$. Let $R(H)$ be an expected reward accumulated along the trajectory H and $p(H;\alpha,\theta)$ be the probability of the trajectory H given the policy and model parameters. Then, the objective function can be expressed as the expected reward over all trajectories

$$J(\alpha,\theta) = \int p(H;\alpha,\theta) R(H) dH.$$

The gradient of $J(\alpha,\theta)$ can be derived using the "log likelihood-ratio trick" [Williams 1992] as:

$$\nabla J(\alpha,\theta) = \int p(H;\alpha,\theta) \nabla \log p(H;\alpha,\theta) R(H) dH. \tag{5}$$

[4]Note that the trajectory is sometimes called the *path* or the *complete history*.

Generally, the gradient in (5) does not have an analytical solution. Hence, the gradient ascent algorithm must estimate the gradient using a Monte Carlo technique. Assuming that the dialogues are numbered $n = 1, \ldots, N$, the equivalent sampled equation is:

$$\nabla J(\alpha, \theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(H_n; \alpha, \theta) R(H_n). \tag{6}$$

By observing that the probability $p(H; \alpha, \theta)$ is a product of probabilities of the model parameters and all actions, observations, and state transitions, the probability of the trajectory H can be defined as:

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

where $p(s_0)$ is the initial state distribution. Since the true state, observation and transition probabilities do not depend on α and θ, it can be shown that the log-gradient has the following form:

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

Then, after substitution of the computed gradient into (6) the resulting gradient has the following form:

$$\nabla J(\alpha, \theta) = \frac{1}{N} \sum_{n=1}^{N} \left[\nabla_\alpha \log p(\tau_n; \alpha)^T, \sum_{t=0}^{T_n - 1} \nabla_\theta \log \pi(a_t^n | b(\cdot | h_t^n; \tau); \theta)^T \right]^T R(H_n).$$

To improve the estimation of the gradient, a constant baseline, B, can be introduced into the equation above. Williams [1992] showed that the baseline does not introduce any bias into the gradient as $\int \nabla p(H; \alpha, \theta) dH = 0$. Nevertheless, if it is chosen appropriately it lowers the variance of the gradient. As a result, the gradient is defined as:

$$\nabla J(\alpha, \theta) = \frac{1}{N} \sum_{n=1}^{N} \left[\nabla_\alpha \log p(\tau_n; \alpha)^T, \sum_{t=0}^{T_n - 1} \nabla_\theta \log \pi(a_t^n | b(\cdot | h_t^n; \tau); \theta)^T \right]^T (R(H_n) - B). \tag{7}$$

The gradient defined in (7) still cannot be used directly since it includes the expected reward $R(H_n)$ which is not usually available. One could approximate the expected reward with a reward observed at the end of each dialogue [Williams 1992]. However, it can be shown that this leads to a biased gradient [Peters et al. 2003]. Another approach is to approximate $R(H_n)$ by a function which is compatible with the policy without biassing the gradient estimate [Sutton et al. 2000; Konda and Tsitsiklis 2000]. A compatible function approximation of $R(H_n)$ parameterised by

a vector $[w_\alpha^T, w_\theta^T]^T$ is as follows:

$$R(H_n) \approx R(H_n; w_\alpha, w_\theta) \approx \left[\nabla_\alpha \log p(\tau_n;\alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta)^T\right] \cdot [w_\alpha^T, w_\theta^T]^T + C \quad (8)$$

To compute the parameters w_α and w_θ, a least squares method can be used. In this case, the expected reward $R(H_n)$ is replaced by the reward r_n observed at the end of each dialogue and the following set of equations is solved whilst minimizing the sum of the squares of the errors:

$$r_n = \left[\nabla_\alpha \log p(\tau_n;\alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta)^T\right] \cdot [w_\alpha^T, w_\theta^T]^T + C \quad (9)$$

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

Similar to the Natural Actor Critic algorithm, the solution $[w_\alpha^T, w_\theta^T]^T$ is the natural gradient which can be used to iteratively improve the policy parameters and the prior of the dialogue model parameters: $\theta' \leftarrow \theta + \beta_\theta w_\theta$, $\alpha' \leftarrow \alpha + \beta_\alpha w_\alpha$. This follows since after applying the approximation of the expected reward (8) to the "plain" gradient (7):

$$\nabla J(\alpha,\theta) \approx \left(\frac{1}{N}\sum_{n=1}^{N}\left[\nabla_\alpha \log p(\tau_n;\alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta)^T\right]^T \cdot \left[\nabla_\alpha \log p(\tau_n;\alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta)^T\right]\right) \cdot [w_\alpha^T, w_\theta^T]^T \quad (10)$$

where the expression in parenthesis is in fact an estimate of the Fisher Information matrix.[5] Thus, equation (10) can be written as $\nabla J(\alpha,\theta) \approx F_{\alpha,\theta}[w_\alpha^T, w_\theta^T]^T$. As a result, the natural gradient of the expected reward is:

$$\widetilde{\nabla} J(\alpha,\theta) = F_{\alpha,\theta}^{-1} \nabla J(\alpha,\theta) \approx F_{\alpha,\theta}^{-1} F_{\alpha,\theta}[w_\alpha^T, w_\theta^T]^T = [w_\alpha^T, w_\theta^T]^T$$

To sum up, NABC solves the least square problem given in (9) to obtain the natural gradient, $[w_\alpha, w_\theta]^T$, of the expected reward $J(\alpha,\theta)$. Then the natural gradient is used to iteratively improve the policy 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 algorithm has to compute the model parameters given the prior. There are two possible ways: the first option is to compute the expected values for the model parameters τ given the distribution $p(\tau;\alpha)$ to provide the new estimates for τ, the second option is to

[5]Note that in the equation (10), the baseline B was cancelled by the constant C. As it can be shown that C is an optimal constant baseline which minimize the variance of the gradient [Williams 1992].

sample the model parameters from the prior[6]. Since informal experiments suggest that both strategies are comparable, only the variant of the algorithm using the expected model parameters is used and evaluated in Section 5. The complete NABC algorithm is described in Algorithm 1.

The NABC algorithm is centred around least square regression. In Section 5.4, the NABC algorithm is used to estimate 1356 policy parameters and 577 model parameters. However, ordinary least square regression becomes unstable when an approximation with hundreds of parameters has to be estimated. For that reason, some form of regularization is needed. In practice, a simple ridge regression with the ridge parameter $\lambda = 0.02$ seems to be sufficient [Hoerl and Kennard 1970].

4.2 Natural Belief Critic

The NABC algorithm described in the previous section assumes that the dialogue system uses a trainable stochastic policy and both the dialogue model and policy parameters are trained jointly. However, it is also useful to be able to train the model parameters separately from the policy parameters. For example, such an algorithm could be used to train the model parameters of a dialogue system using a hand-crafted policy such as the one described in Section 2.3. The Natural Belief Critic (NBC) algorithm is a variant of NABC, which assumes that the policy is fixed. The main benefit of NBC is that it can be used to optimise the model parameters of a dialogue system using an arbitrary policy [Jurčíček et al. 2010].

Since the policy is fixed during training and only the prior of the model parameters needs to be optimised, only the natural gradient of $J(\alpha, \theta)$ w.r.t α needs to be estimated. Given the fact that $\nabla_\alpha \log p(H; \cdot, \alpha) = \nabla \log p(\tau; \alpha)$, NBC solves the following set of equations:

$$r_n = \nabla_\alpha \log p(\tau_n; \alpha)^T \cdot w + C \quad \forall n \in \{1, \ldots, N\} \tag{11}$$

to obtain the natural gradient w of the expected reward $J(\cdot, \alpha)$. The complete NBC algorithm is described in Algorithm 2.

The NBC algorithm can also be understood as a random search algorithm. A typical random search algorithm maximizing a function F is an iterative algorithm with two steps in each iteration. First, it draws a set of samples, $x_1, \ldots, x_N$, from some probability distribution centred around the current point estimate of the maximizer of the function F. Second, it uses some update rule to compute a new estimate of the maximizer from the evaluations of the function F at $x_1, \ldots, x_N$. Sometimes, a random search algorithm also performs adaptation of parameters of the sampling distribution.

One can see that NBC fits the procedure described above. First, the NBC algorithm samples from the prior for the model parameters. Second, NBC updates the parameters of the prior by adding a multiple of the natural gradient computed from the evaluations of the sampled dialogue model parameters. Note that by updating the prior parameters, the NBC algorithm also updates the point estimate of the maximizer. In the case of NBC, the point estimate of the model parameters is the

[6]In practice, it is better to sample the model parameters continuously when running the dialogue manager to avoid the risk of sampling unlikely model parameters and never using likely model parameters.

Algorithm 1 Natural Actor and Belief Critic

Let τ be the parameters of the dialogue model
Let $p(\tau;\alpha)$ be a prior for τ parameterised by α
Let $\pi(a_t|b_t;\theta)$ be a policy parameterised by θ
Let I be number of executed iterations
Let N be a number of episodes sampled in each iteration
Let T be number of turns in a dialogue
Let β_α and β_θ be step sizes
Input: α_1 - initial parameters of the prior for τ
Input: θ_1 - initial parameters of the policy π
Output: θ - the policy parameters
Output: τ - the dialogue model parameters

for $i = 1$ to I **do**
  `Collecting statistics:`
  for $n = 1$ to N **do**
    Draw parameters $\tau_n \sim p(\cdot;\alpha_i)$
    `Execute the episode:`
    for $t = 0$ to $T_n - 1$ **do**
      Draw action $a_t^n \sim \pi(\cdot|b(\cdot|h_t^n);\theta_i)$
      Observe the reward r_t^n
      Observe o_{t+1}^n
      $h_{t+1} \leftarrow h_t \cup \{a_t^n, o_{t+1}^n\}$
    end for
    Record $r_n = \sum_{t=0}^{T_n-1} r_t^n$
  end for
  `Critic evaluation:`
  Choose $[w_{\alpha,i}, w_{\theta,i}]^T$ to minimize the sum of the squares of the errors of

$$r_n = \left[\nabla_\alpha \log p(\tau_n;\alpha_i)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta_i)^T\right] \cdot [w_{\alpha,i}^T, w_{\theta,i}^T]^T + C \quad \forall n \in \{1,\ldots,N\}$$

  `Parameter update:`
  $\alpha_{i+1} \leftarrow \alpha_{i+1} + \beta_\alpha w_{\alpha,i}$
  $\theta_{i+1} \leftarrow \theta_{i+1} + \beta_\theta w_{\theta,i}$
end for

$\theta \leftarrow \theta_{I+1}$
$\tau \leftarrow \int p(\tau;\alpha_{I+1})\tau d\tau$

expectation of the model parameters given the prior.

Thus, later in Section 5.5, the NBC algorithm is compared to other state-of-the-art random search techniques such as SPSA [Spall 2003] and CMA-ES [Hansen and Ostermeier 2001]. However, SPSA and similar techniques have strong requirements on the generation of the evaluated samples and they are affected by scaling of the parameters [Peters and Schaal 2008b]. During informal experiments, it was

Algorithm 2 Natural Belief Critic

Let τ be the parameters of the dialogue model
Let $p(\tau;\alpha)$ be a prior for τ parameterised by α
Let N be the number of dialogues sampled in each iteration
Let I be the number of training iterations
Let β be a step size
Input: α_1 - initial parameters of the prior for τ
Input: π - a fixed policy
Output: τ - the dialogue model parameters

for $i = 1$ to I **do**
    `Collecting statistics:`
    for $n = 1$ to N **do**
        Draw parameters $\tau_n \sim p(\cdot;\alpha_i)$
        Execute the dialogue according the policy π
        Observe the reward r_n
    end for
    `Critic evaluation:`
    Choose w_i to minimize the sum of the squares of the errors of
        $r_n = \nabla_\alpha \log p(\tau_n;\alpha_i)^T \cdot w_i + C$
    `Parameter update:`
    $\alpha_{i+1} \leftarrow \alpha_i + \beta w_i$
end for

$\tau = \int p(\tau;\alpha_{I+1})\tau d\tau$

observed that SPSA and CMA-ES were not able directly optimize the prior parameters α. However, it was noticed that they can be used in a transformed parameter space. One viable transformation is, for example, $t(\alpha) : \alpha \rightarrow \log \alpha$.

4.3 The dialogue model parameters prior

In order to use NABC or NBC in practice, a prior for the model parameters τ is needed. Since the parameters of the BN described in Section 2.2 are parameters of multiple multinomial distributions, a product of Dirichlet distributions provides a convenient prior.

Formally, for every node $j \in \{1,\ldots,J\}$ in the BN, 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: $\tau_j = \left[\tau_{j,1},\ldots,\tau_{j,|par(j)|}\right]$. 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 $\tau = [\tau_1,\ldots,\tau_J]$ be a vector of all parameters in the BN. Then, the probability $p(\tau;\alpha)$ from (11) can be

defined as:

$$p(\tau;\alpha) = \prod_{j=1}^{J} \prod_{k=1}^{|par(j)|} Dir(\tau_{j,k};\alpha_{j,k}). \quad (12)$$

After taking the log-derivative of (12) w.r.t. α, the following gradient is obtained:

$$\nabla_\alpha \log p(\tau;\alpha) = \sum_{j=1}^{J} \sum_{k=1}^{|par(j)|} \nabla_\alpha \log Dir(\tau_{j,k};\alpha_{j,k})$$

which has a closed form solution and can be used in (9) or (11) to compute the natural gradient.

4.4 Plain gradient algorithms

To establish the importance of using the natural gradient (see Section 5), the previously proposed algorithms will be compared to methods using the "plain" gradient. This section describes these algorithms for the two previously described tasks:

(1) joint estimation of the model and the policy parameters,
(2) estimation of the model parameters only.

In the first case, one can directly use the gradient defined by (10). This follows since after the estimation of the parameters of the approximation of the expected cumulative reward (8), the "plain" gradient (7) can be computed. The algorithm has a very similar structure to the NABC algorithm. The only difference is that after estimation of the parameters of the expected cumulative reward (8), the "plain" gradient (10) has to be computed. Note, that the "plain" gradient has to be used to update the policy and the prior of the dialogue model parameters. This algorithm will be denoted as PGTAB.

In the second case, the "plain" gradient can be simply derived from (10) by neglecting the part related to the policy parameters. As a result the "plain" gradient of the expected cumulative reward w.r.t. the parameters of the prior is as follows:

$$\nabla_\alpha J(\alpha;\cdot) \approx \left(\frac{1}{N} \sum_{n=1}^{N} \nabla_\alpha \log p(\tau_n;\alpha) \nabla_\alpha \log p(\tau_n;\alpha)^T \right) w_\alpha. \quad (13)$$

Again, the algorithm has a similar structure to its counterpart - the NBC algorithm. There is only one extra step: after the estimation of the parameters of the expected cumulative reward (11), the "plain" gradient is computed using (13). This algorithm will be denoted as PGTB.

Note that the two algorithms described above are based on the policy gradient theorem (PGT) [Sutton et al. 2000].

5. EVALUATION

An experimental evaluation of the NABC and NBC algorithms was conducted using the BUDS dialogue system described in Section 2. The goal of the evaluation was to test whether NABC and NBC could estimate a set of dialogue model parameters which yielded better performance than a set of carefully hand-crafted model parameters and parameters obtained by several baseline algorithms.

The evaluation was in two parts:

(1) A set of model and policy parameters were estimated jointly by the NABC algorithm and compared to the performance obtained using:
   (a) the hand-crafted model parameters and a stochastic policy trained by the eNAC algorithm,
   (b) jointly estimated policy and model parameters trained by the PGTAB algorithm using the "plain" gradient.
(2) A set of model parameters were estimated by the NBC algorithm using a finely tuned hand-crafted policy. The result was compared to the performance of:
   (a) the hand-crafted model parameters and the hand-crafted policy referred as to HDC,
   (b) the model parameters trained by the PGTB algorithm using the "plain" gradient,
   (c) the model parameters estimated by the SPSA and CMA-ES random search algorithms.

The systems were trained and tested using an agenda-based user simulator [Schatzmann et al. 2005], for the Town-Info domain, which provides tourist information for an imaginary town [Thomson and Young 2010; Young et al. 2010]. The user simulator incorporates a semantic concept confusion model, which enables the systems to be trained and tested across a range of semantic error rates.

The basic idea of the confusion model is that the correct dialogue act is randomly placed in the N-best list given the error rate used for evaluation. First, an N-best list of confidence scores is sampled from a Dirichlet distribution. Then, the position of the correct dialogue act is determined by sampling from the multinomial distribution corresponding to the sampled confidence scores. Finally, the rest of the N-best list is filled with automatically generated incorrect hypotheses. The parameters of the Dirichlet distribution are set to place a correct dialogue act on the top of the N-best list with probability $1-e$ and to omit the correct dialogue act from the N-best list with probability $1-e^2$, where e is the error rate. Although the NABC and NBC algorithms are evaluated on error rates between 0% to 50%, comparison of the performance between different parameter settings should be mainly done on error rates between 30% to 40% since the average top semantic error rate obtained in previous real user trials was about 34% [Thomson and Young 2010; Young et al. 2010].

5.1 Reward function

The reward function used in the evaluation experiments 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.

For training, the above reward function is not appropriate since in the first iteration, the policy is sampling its actions from a uniform distribution. Hence, the

chance of a successful dialogue is very low, causing the eNAC algorithm to optimise only the length of each dialogue instead of optimising both the length and the success rate. This arises because it is locally more efficient to say "goodbye" in the first turn than to try to succeed with a longer dialogue. As a result, the policy converges to a local optimum corresponding to very short dialogues which are all unsuccessful.

To alleviate this problem, it is better to optimise only the success rate at the beginning of the training and later, as the success rate of the policy increases, also optimise the length of dialogues. Thus, the eNAC algorithm uses a reward function which still awards 20 for a successful dialogue and 0 for unsuccessful ones. However, it assigns 0 for each turn when the policy success rate is 0 and it linearly increases the per turn penalty to -1 as the success rate tends towards 100%.

5.2 Dialogue model for the Town-Info domain

The Bayesian Network for the Town-Info domain contains nine sub-goals: name of the venue, type of venue, area, price range, nearness to a particular location, type of drinks, food type, number of stars and type of music. Every sub-goal has a corresponding sub-history node. The network also has nodes to represent address, telephone number, a comment on the venue and the price. However, for these only their sub-history nodes are used since a user can only ask for values of these slots and cannot specify them as query constraints. Finally, the network includes the "method" and "discourse" nodes which were described in Section 2.2. Although the dialogue manager does not ask about these nodes explicitly, their values are inferred just like any other node.

The history, "method", and "discourse" nodes use fully parameterised conditional probabilities in order to capture the detailed characteristics of dialogue flow. All of the other sub-goal nodes use parameter tying as described in Section 2.2. Overall this results in a total of 577 parameters in the dialogue model.

5.3 The policy actions

The BUDS policy implements 24 slot-level summary actions: 3 actions ("request", "confirm", "select") for each sub-goal apart from the "name of the venue" sub-goal. The system is not allowed to ask a user for the name of a venue as the goal of the dialogue is to find a suitable venue and provide the name.

The policy implements 8 global summary actions: inform about a venue matching the query constraints, inform about a venue specified by a name[7], inform about an alternative venue, provide more details about the last informed venue, inform about requested slots, inform about slots being confirmed by a user, repeat the last prompt, ask if a user wants anything more, end the dialogue.

The features for the slot-level and the global actions are described in Section 2.3. Overall there are 1356 parameters in the stochastic policy for the slot-level and the global summary actions.

[7]This is a response in a scenario when a user is looking for information about a venue with a specific name, e.g. "Do you have the phone number for Charlie Chan?" instead of a venue matching some constraints, e.g. "Can you give me the phone number of any cheap Chinese restaurant?"

5.4 NABC experiments

In the first experiment, the NABC algorithm was used to jointly estimate a set of dialogue model and policy parameters by running the algorithm for 120 iterations. The user simulator was set to produce dialogues on randomly selected error rates. The error rate for each simulated dialogue was sampled from the uniform distribution at the interval $[0\%, 50\%]$. The total number of sampled dialogues per iteration was 32k. Both the policy parameters and the parameters of the prior of the dialogue model were initialised by uninformative (uniform) parameters. In total, there were 1933 parameters to be estimated. The baseline system was composed of a stochastic policy trained by the eNAC algorithm. In this case, only the policy was initialised by uninformative parameters as the dialogue model used the hand-crafted parameters. Both the baseline system and the system with the learnt BN and policy parameters were evaluated over error rates ranging from 0% to 50%. At each error rate, 5000 dialogues were simulated and to reduce the variance of results, this training and evaluation procedure was executed 5 times. The averaged results along with 95% confidence intervals are depicted in Figure 3. The results show that the system with the model and policy parameters trained using NABC significantly outperforms the baseline system with hand-crafted model parameters and the policy parameters trained using the eNAC algorithm at error rates above 15% ($p < 0.05$). For example, at 35% error rate, the mean reward was increased by 0.96 absolutely from 7.07 to 8.03.

Figure 4 compares the learning curves of the baseline system with hand-crafted model parameters and a policy trained using the standard eNAC algorithm, and a system for which both the model and the policy parameters have been jointly trained using the NABC algorithm. At the beginning the eNAC 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. Figure 4 also compares the learning curves of the NABC algorithm and the PGTAB algorithm which uses the "plain" gradient. The NABC algorithm converges significantly faster than the PGTAB algorithm. The results also suggests that the use of the natural gradient is necessary as the performance of the system with parameters estimated by the PGTAB algorithm does not reach the performance of the system with hand-crafted model parameters and a policy trained by the eNAC algorithm. The plotted mean reward is the average mean reward over the error rates 0% to 50% used during training. Informal experiments suggest that the NABC and eNAC systems are fully trained since further increasing the number of dialogues used per iteration does not improve the performance. After 120 iterations amounting to 3,840,000 dialogues, both the model and the policy parameters converge.

Although the natural gradient should be invariant to linear parameter transformation, the experiments suggests that the step size for the parameters of the prior of the model parameters, β_α, must be significantly higher than the step size for the policy parameters, β_θ. It was observed that the policy reduces its variance much faster than the prior of the model parameters when the step size is the same for both parts of the gradient. Presumably the parameterisation of the prior of the model parameters modelled as a product of Dirichlet distributions is non-linearly different

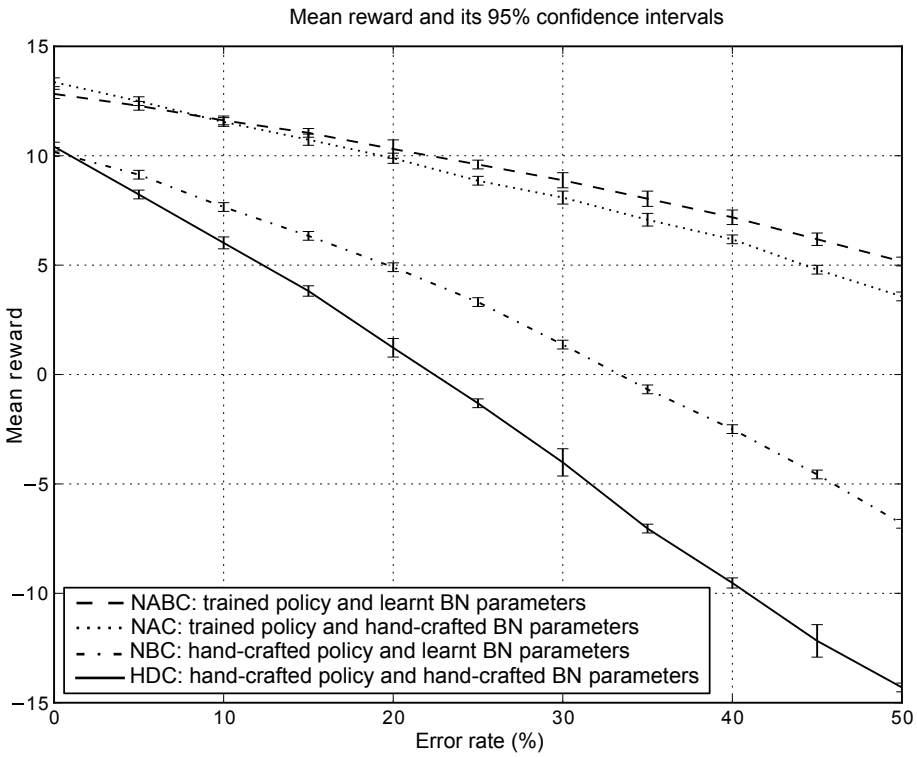

Fig. 3. Results for the NABC and NBC algorithms. First, it compares the mean rewards of the hand-crafted BN model parameters and the model parameters learnt by the NABC algorithm when using the trained policy. Second, it compares the mean rewards of the hand-crafted BN model parameters and the model parameters learnt by the NBC algorithm when using the fixed hand-crafted policy.

to the parameterisation of the policy modelled as a softmax function. Setting a much lower step size for the policy parameters in the NABC algorithm prevents the policy from lowering its variance too quickly and enables it to adapt to the changed model parameters. Based on some initial experiments, the step size for the model parameters was set to be fifty times higher than the step size for the policy parameters. The actual values used for the step sizes are dependent on the definition of the reward function. For example, if the reward function awarded 200 for a successful dialogue then the gradient would be 10 times larger than the gradient computed using the original reward function defined in Section 5.1. For that reason, the optimal step sizes have to be tuned and tested on a particular reward function. It is therefore important to keep in mind that the two step sizes β_α and β_θ can differ significantly.

5.5 NBC experiment

In the second experiment, the dialogue model parameters were estimated by the NBC algorithm using the hand-crafted policy. The setup of the experiment was

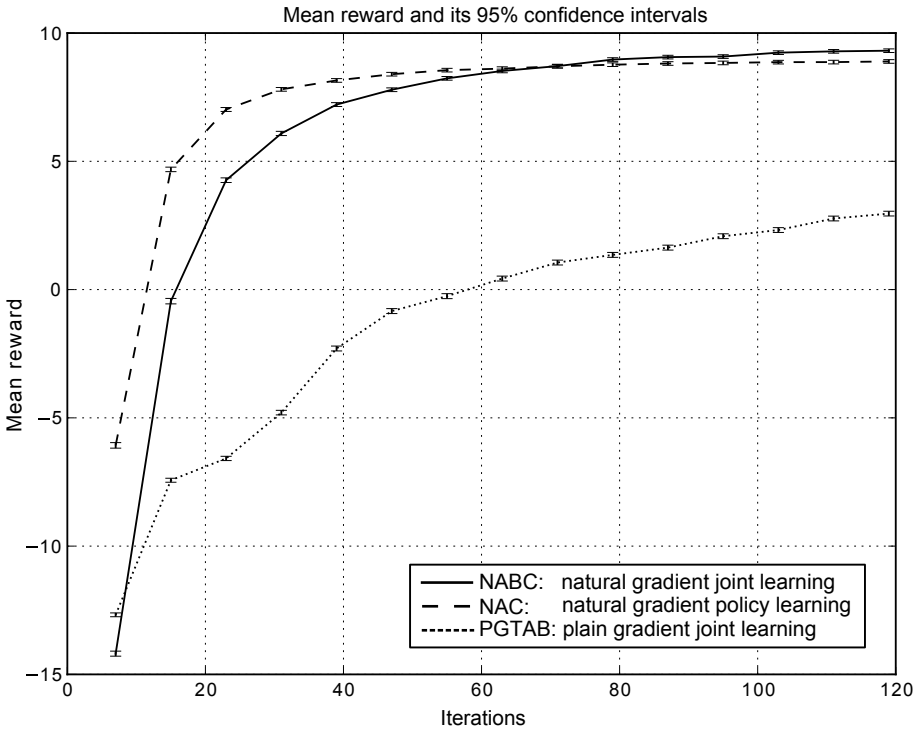

Fig. 4. Comparison of average reward across all error rates during training of the baseline system using the eNAC algorithm to train the policy and the hand-crafted model parameters, the system using the NABC algorithm to train both the model and the policy, and the system using the PGTAB algorithm to train both the model and the policy.

very similar to the previous experiment in Section 5.4. The NBC algorithm was executed for 120 iterations with the simulator set to produce dialogues with error rates between 0% and 50%. The error rates were again uniformly distributed among the dialogues. In each iteration, 16k dialogues were sampled. The prior of the dialogue model was initialised by uninformative parameters. There were 577 dialogue model parameters to be estimated. For the baseline system, no parameter training was necessary as both the model parameters and the policy were hand-crafted. Both the baseline system and the system with the learnt BN parameters were evaluated over error rates ranging from 0% to 50%. At each error rate, 5000 dialogues were simulated and to reduce the variance of results, this training and evaluation procedure was executed 5 times. The averaged results along with 95% confidence intervals are depicted in Figure 3. The results show that the system with trained BN parameters significantly outperforms the system with hand-crafted parameters especially at high error rates. For example, at 35% error rate, the mean reward was increased by 6.36 absolutely from -7.04 to -0.68. Inspection of the results suggests that this improvement can be mostly attributed to the sub-optimality of the hand-crafted policy and the ability of the learnt BN parameters to compensate for this.

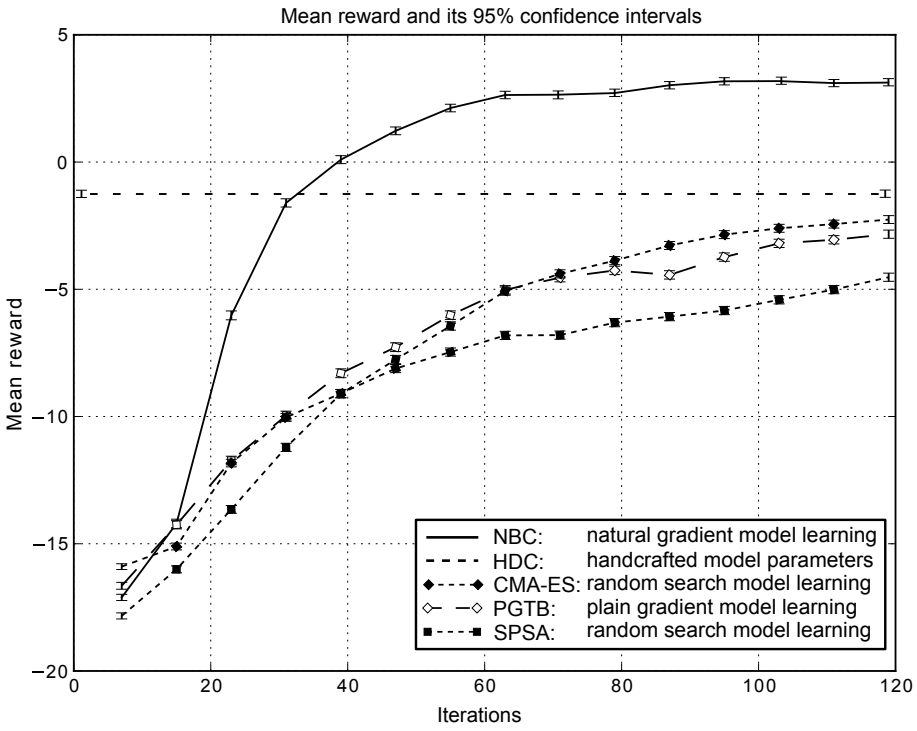

Fig. 5. Comparison of average reward across all error rates during training of the model parameters using the NBC, PGTB, SPSA, and CMA-ES algorithms.

In Section 4.2, it was suggested that the NBC algorithm can be seen as a random search algorithm. For that reason, the algorithm was compared to two state-of-the-art random search algorithms: SPSA [Spall 2003] and CMA-ES [Hansen and Ostermeier 2001]. To assess the importance of the natural gradient, the NBC algorithm was also compared to the PGTB algorithm. Note that the PGTB algorithm is using the "plain" gradient when updating the prior parameters. The results, plotted in Figure 5, show that the NBC algorithm converges significantly faster then any of the compared algorithms. The lower performance of PGTB can be attributed to the use of the "plain" gradient.

CMA-ES is the best of the alternative algorithms; however, it appears that the performance of the parameters estimated by this algorithm levels off before it reaches the performance of the handcrafted parameters (referred to as HDC).

The lowest learning rate was achieved by the SPSA algorithm. SPSA is popular for its simplicity; however, the performance of the algorithm is highly impacted by sampling process of perturbations in the parameters. When it is used to optimise problems where the impact of changes in the parameters is large, the algorithm is not very efficient and the learning process can even diverge [Peters and Schaal 2008b]. Also, the algorithm does not take advantage of the knowledge of the gradient of the prior when compared to PGTB. Consequently, the estimates of the gradient have much higher variance and therefore a smaller step size must be used

to prevent divergence of the algorithm.

5.6 Efficiency analysis

To formally analyse the computational complexity of the NABC algorithm, consider that the algorithm is running for I iterations and N dialogues are sampled per iteration. Further, suppose that the complexity of simulating one dialogue is T. Consequently, this results in complexity $O(NT)$ for sampling N dialogues in one iteration. In each iteration, the NABC algorithm involves solving a least square problem which has complexity $O(NP^2 + P^3)$ were P is the number of parameters being estimated. So, the over all complexity of the NABC algorithm is $O(I(NT + NP^2 + P^3))$. Similar results can be obtained for the NBC algorithm and also for the versions of the algorithms using the "plain" gradient.

Although the complexity of the algorithm is cubic in the number of parameters being estimated, P, in practice this is not the main contributor to the total running time. The experiments described in the previous sections were performed on a grid of computers with 16 machines each having two Intel Xeon 2.8GHz processors with 4 cores and 24GB of RAM each. In total, there were 128 cores used during training. In each iteration, all 128 cores were used to simulate dialogues and the simulations finished in about 15 minutes on average. Once the simulations were done, the least squares problem was solved in less than 10 seconds on one core. The total computation time for the NABC algorithm for the experiment in Section 5.4, was about 30 hours - 120 iterations multiplied by 15 minutes.

When taking into account the number of cores, the simulations took $128 \cdot 15 = 1920$ minutes of computer time for one iteration. If this time is compared to the time needed for solving the regression problem, then the time consumed by the regression problem appears to be negligible. Therefore, to evaluate the efficiency of the NABC and NBC algorithms, one has to mainly pay attention to the number of dialogues needing to be sampled to estimate an accurate gradient and the total number of iterations to reach a local optimum.

Figure 4 compares the learning curves of the NABC algorithm and the "plain" gradient PGTAB algorithm. The results suggest that NABC converges significantly faster than the ABC algorithm. It has already been noted in the literature that the use of the natural gradient can converge up to 3 orders of magnitude faster than methods using "plain" gradients [Peters and Schaal 2008a]. Although the methods using the "plain" gradient eventually reach a local optimum, it can just take too long. To test whether the PGTAB algorithm does converge, the training using the PGTAB algorithm continued for further 80 iterations. This gave an improvement of about 2.1, still some way away from the optimum. No further iterations were performed, since continuation to convergence would be prohibitively time consuming.

The space complexity of the NABC and NBC algorithms is mainly defined by the least square method. The least square method involves the construction of a square matrix and an inversion of that matrix. In the experiments, square matrices of order 1,933 or 577 had to be inverted and the inversion of the matrices took less than 1 second. As the inversion has quadratic complexity in space and cubic complexity in time, the largest square matrix which can be inverted on the used grid computers using NumPy [Oliphant 2007] is of order 20,000 and takes about

70 minutes. When the inversion is parallelised over 8 cores on one machine than it takes about 10 minutes. Although some extra optimizations can be performed, the NABC and NBC algorithms are clearly limited by the use of the least square method. Nevertheless, the algorithms would enable scaling from the current task with 9 slots to a task with up to 100 slots. However, note that at least 10 times more dialogues would have to be sampled in each iteration to train such a system.

6. DISCUSSION

As noted in Section 1, the task of learning dialogue model parameters is not a new problem and there have been several attempts in this direction. The most relevant works are: Georgila et al. [2005], Kim et al. [2008], Williams [2008b], Doshi and Roy [2007], Syed and Williams [2008] and Thomson [2010].

For example, Georgila et al. [2005] and Kim et al. [2008] used maximum likelihood estimates from an automatically annotated corpus of real dialogues. This approach is based on the assumption that the automatically annotated states are correct. The method is problematic as the quality of the estimated model depends on the quality of the automatic annotation which itself relies on having a good dialogue model.

Williams [2008b] obtained maximum likelihood estimates from a corpus with manually annotated dialogue states. Although this approach can provide good estimates of the dialogue model, it is very laborious and in practice only a small number of dialogues can be obtained. Also, in many real dialogues, some components of the dialogue state, especially the user's goal, are hard to determine. Hence, this approach is usually restricted to cases where the user's goal remains constant and the dialogue is simple to annotate.

Doshi and Roy [2007] presented a dialogue model which was trained by Viterbi learning [Rabiner 1989] using a reinforcement signal based on the successful completion of each dialogue. The dialogue model had one slot with 7 values and accepted 19 different observations. Every time the dialogue system succeeded in a dialogue, the most likely state sequence decoded by the Viterbi algorithm was used to update the probabilities of the dialogue model. By conditioning on the successful completion of a dialogue, it was assumed that the decoded most likely sequence of states was correct. In this method, the initial model parameters have to be handcrafted with some reasonable quality to be able to improve the parameters. From the presented work, it is not clear whether it scales to dialogue models with multiple slots with tens or hundreds of values.

Syed and Williams [2008] showed how to use Expectation-Maximization (EM) to learn parameters of an observation model. A dialogue model can be factored into a transition model between states, and an observations model, but in this work, no attempts were made to learn the transition model. Instead, it was assumed the goal remains constant and hence the complexity of the problem was significantly reduced and a tractable method was achieved.

Thomson [2010] used Expectation-Propagation (EP) to infer hidden state information together with the model parameters. The algorithm treats the task of learning the model parameters in an unsupervised manner. However, it is not clear to what extent likelihood maximisation over a dialogue corpus correlates with the

expected cumulative reward of the dialogue system.

One of the requirements of the methods based on maximizing likelihood of the data, such as EM and EP, is that the model must be generative. In other words, the observations must be conditioned on the dialogue state. The belief update (1) was transformed into a generative model by using Bayes theorem, and although this form is convenient for using with EM and EP, it is not necessary. Instead, $b(s_t|h_t;\tau)$ can be modelled directly by optimising a target evaluation metric and this is a potentially interesting future direction. Even if the model is generative, it is still useful to directly optimise the evaluation metrics. Already in the speech community, discriminative training which optimises the parameters of a hidden Markov model to maximise word accuracy has achieved significant results [Kapadia 1998]. One of the reasons is that the structure of generative models is not perfect due to many approximations. Consequently, additional optimisation of the parameters can improve their performance. Nevertheless, the EM and EP techniques are important as they enable the models to be trained in an unsupervised manner and utilise corpora of real dialogues.

To avoid the problem of building an estimator of a belief state and eventually learning the parameters of the estimator, the reinforcement community has developed several techniques for solving POMDPs based on policies with internal memory[8]. These techniques learn to map observations directly to actions and they use their internal memory to summarise important information from the past observations. For example Wierstra et al. [2010], used recurrent neural networks (RNN), to approximate the policy. At each step the RNN updates its internal memory and proposes a new system action based on the accumulated information in the internal memory and the last observation. Conveniently, the RNN is designed in such way that it is possible to compute the gradient w.r.t. all parameters of the RNN. As result, the parameters of the RNN can be learnt by gradient descent. Another similar approach is the work of Aberdeen [2003]. In this work, the memory is represented as a finite set of internal states and the policy maintains beliefs over all of these states. Every time there is a new observation, the policy updates its beliefs and proposes a new action. Again, the gradient w.r.t. all parameters can be obtained and gradient descend can be used. Although such methods may seem to be attractive, they are not suitable for spoken dialogue systems. In both examples, there is no meaning associated with the values in the internal memory. The use of internal memory is only optimised to maximise the expected cumulative reward of the final policy. However, a dialogue system needs an explicit model of dialogue state. For example, the estimate of the most likely dialogue state is used to query a database and the obtained information is used in the system actions.

In Sections 5.4 and 5.5, the evaluation of two algorithms for estimation of parameters in spoken dialogue systems was described. The NABC algorithm jointly estimates the dialogue model and the policy parameters whereas the NBC algorithm estimates only the model parameters assuming that the policy is fixed. Both these algorithms directly optimise the target evaluation metric - the expected cumulative reward and they do so without any of the limitations of the alternatives discussed above.

[8]The internal memory is sometimes called internal state.

The system trained by the NABC algorithm significantly outperforms the handcrafted model parameters and the eNAC trained policy. Also, the NABC algorithm outperforms the PGTAB algorithm. The results show that the use of the natural gradient is critical as the learning rate of the PGTAB algorithm is too slow to outperform the handcrafted model parameters in some reasonable time. The main benefit of the NABC algorithm is that a developer of dialogue systems can avoid the cost of hand-crafting the parameters while gaining increased performance.

For cases where there is an existing fixed policy, perhaps prescribed by other design considerations, the NBC algorithm can significantly improve performance by optimising the model parameters. It is presumed in this case that the sub-optimality of the hand-crafted policy is compensated by the learnt BN parameters. Thus, if the system designer is required to use a policy with a certain fixed behaviour then optimising the model parameters with NBC can deliver a significant gain. Since the NBC algorithm can be understood as a random search algorithm, it was also compared to other state-of-the-art techniques such as SPSA and CMA-ES. The results showed that the learning rate of the NBC algorithm is significantly faster. Similarly to NABC, the NBC algorithm is also faster that the PGTB algorithm which is using the "plain" gradient.

The model parameters estimated by NBC are significantly different to the model parameters learnt by NABC. When the NBC algorithm estimates the natural gradient, it does not take into account the actions taken by the policy and the effects of the actions are simply averaged. On the other hand, the NABC algorithm uses information about the actual actions taken. For example, if in NABC unlikely actions are sampled and the resulting dialogue obtains a low reward, the cause of the low reward is attributed to those actions and not to the sampled model parameters. Similarly, if NABC samples unlikely model parameters and the resulting dialogue obtains a high reward, then the gain is attributed to the model parameters rather than the sampled actions. Thus, the NABC algorithm actively uses the information about sampled actions to lower the variance in the estimate of the gradient of the parameters.

When training both the model and the policy parameters, joint optimisation is necessary. One could think of a training scheme in which the NBC and eNAC algorithms alternate. The NBC algorithm would be used first to improve the model parameters, and the eNAC algorithm would be used second to improve the policy parameters. This two step training would then continue until convergence. Such a training scheme was used in [Jurčíček et al. 2010]. However, in that case it was assumed that there was a reasonable set of handcrafted model parameters which can be used to initialise the training process.

The difficulty with this alternating approach is that neither the policy or the model parameters can be improved when both initialised by uninformative parameters. In the first stage, for example, the policy cannot be improved since the estimate of the belief state is based on uninformative parameters. In other words, the dialogue model does not have any preference for particular states given the last observation and the last actions. The problem is that all dialogue states are equally likely given the last observation and the last action. Consequently, only one belief state is presented to the policy learning algorithm. Similarly, the model parameters

cannot be improved because the actions sampled from the stochastic policy using uninformative parameters are all equally likely. In other words, the policy does not have any preference for particular actions given the belief state when using uninformative parameters.

The methods presented in this paper can be extended in principle to optimise any parameter in the entire dialogue system. The trick is that the learning of non-differentiable parameters of the cumulative reward[9] is replaced by learning differentiable parameters of their prior.

The design of the prior is entirely a matter for the system developer. The main goal should be to capture important correlations between different parameters. In Section 4.3, the prior for the model parameters was designed as a product of Dirichlet distributions. Although such a prior is differentiable and it is easy to sample from such a distribution, it is not necessarily the most efficient prior since it assumes that parameters of a particular BN node are independent of parameters of other BN nodes. For example, information about the likelihood of changing the value of the "food" node could be used to learn about the probability of change in the "drink" node since it would be reasonable to expect that these probabilities are not very different. So, one way to improve the learning rate and reduce the number of sampled dialogues would be to use domain specific priors, which could be done automatically based on some form of ontology [Young et al. 2010].

7. CONCLUSION

This paper has proposed a novel reinforcement learning method for estimating parameters of statistical spoken dialogue systems. The Natural Actor and Belief Critic (NABC) algorithm jointly optimises the model and the policy parameters of a POMDP-based dialogue system. Based on observed rewards obtained in a set of training dialogues, the algorithm estimates the natural gradient of the expected cumulative reward of a dialogue system and then adapts jointly the Dirichlet prior distributions of the model parameters and the stochastic policy so as to maximise the expected cumulative reward. Simulations showed that the performance of the jointly learnt model and the policy parameters significantly outperforms the baseline system. In addition, this paper has presented a variant of the NABC algorithm aimed at training only the model parameters while the policy is fixed. This algorithm, called Natural Belief Critic (NBC), is useful for optimising the performance of a dialogue system which has a hand-crafted policy exhibiting certain required characteristics. Simulations showed that the NBC algorithm learns a significantly better set of model parameters when compared with an initial set of hand-crafted model parameters.

Overall we believe that this class of natural gradient based optimisation algorithms offer considerable potential for improving the performance of statistically based spoken dialogue systems. Furthermore, the algorithms are not limited to the dialogue management component but could in future be extended to optimise parameters in both the speech understanding and speech generation components

[9]Note that the model parameters are differentiable parameters of the likelihood function over a corpus of dialogues; however, the model parameters are non-differentiable parameters of the cumulative reward.

with respect to the required reward function.

APPENDIX

A. THE GRADIENT OF THE LOGARITHM OF THE SOFTMAX POLICY

To derive the gradient of the logarithm of the softmax policy, $\nabla_\theta \log \pi(a_t|b(\cdot|h_t;\tau);\theta)$, first recall that the softmax policy (2) is:

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

Note that θ are the policy parameters and $\Phi_a(b)$ maps the belief state, b, into a vector of features. For the sake of simplicity, denote $b(\cdot|h_t;\tau)$ as b and a_t as a. Then the gradient $\nabla_\theta \log \pi(a|b;\theta)$ can be derived as follows:

$$\begin{aligned}
\nabla_\theta \log \pi(a|b;\theta) &= \nabla_\theta \log \left(\frac{e^{\theta^T \cdot \Phi_a(b)}}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)}} \right) \\
&= \nabla_\theta \log \left(e^{\theta^T \cdot \Phi_a(b)} \right) - \nabla_\theta \log \left(\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)} \right) \\
&= \Phi_a(b) - \frac{1}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)}} \nabla_\theta \left(\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)} \right) \\
&= \Phi_a(b) - \frac{1}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)}} \left(\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)} \Phi_{\tilde{a}}(b) \right) \\
&= \Phi_a(b) - \sum_{\hat{a}} \frac{e^{\theta^T \cdot \Phi_{\hat{a}}(b)}}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b)}} \Phi_{\hat{a}}(b)
\end{aligned}$$

ACKNOWLEDGMENTS

This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSIC project: www.classic-project.org). The authors would like to thank Simon Keizer, for his work on the user simulator used here, as well as Milica Gašić, François Mairesse, Kai Yu, and Jorge Prombonas for their useful comments and discussions.

REFERENCES

Aberdeen, D. A. 2003. Policy-gradient algorithms for partially observable Markov decision processes. Ph.D. thesis, Australian National University.

Amari, S. 1998. Natural gradient works efficiently in learning. *Neural Computation 10,* 2, 251–276.

Bishop, C. 2006. *Pattern Recognition and Machine Learning.* Springer.

Bui, T. H., Poel, M., Nijholt, A., and Zwiers, J. 2009. A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. *Natural Language Engineering 15,* 2 (March), 273–307.

Doshi, F. and Roy, N. 2007. Efficient model learning for dialog management. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, New York, NY, USA, 65–72.

Engel, Y., Mannor, S., and Meir, R. 2005. Reinforcement learning with Gaussian Processes. In *Proceedings of the 22nd International Conference on Machine Learning*. Bonn, Germany.

Geist, M., Pietquin, O., and Fricout, G. 2009. Kalman Temporal Differences: the deterministic case. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning ADPRL 2009*. Nashville, TN United States, 185–192.

Georgila, K., Lemon, O., and Henderson, J. 2005. Automatic annotation of COMMUNICATOR dialogue data for learnin dialogue strategies and user simulations. In *In Ninth Workshop on the Semantics and Pragmatics of Dialogue*.

Hansen, N. and Ostermeier, A. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9,* 2, 159–195.

Hoerl, A. E. and Kennard, R. W. 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics 12*, 55–67.

Jurčíček, F., Thomson, B., Keizer, S., Gašić, M., Mairesse, F., Yu, K., and Young, S. 2010. Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In *Submitted to Interspeech 2010*.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence 101*, 99–134.

Kapadia, S. 1998. Discriminative training of hidden Markov models.

Kim, D., Sim, H. S., Kim, K., Kim, J. H., Kim, H., and Sung, J. W. 2008. Effects of User Modeling on POMDP-based Dialogue Systems. In *Proceedings of Interspeech*.

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

Oliphant, T. E. 2007. Python for scientific computing. *Computing in Science and Engineering 9,* 3, 10–20.

Peters, J. and Schaal, S. 2008a. Natural actor-critic. *Neurocomput. 71,* 7-9, 1180–1190.

Peters, J. and Schaal, S. 2008b. Reinforcement learning of motor skills with policy gradients. *Neural Networks 21,* 4, 682 – 697. Robotics and Neuroscience.

Peters, J., Vijayakumar, S., and Schaal, S. 2003. Reinforcement Learning for Humanoid Robotics. In *Proc. of the Third IEEE-RAS International Conference on Humanoid Robots*.

Peters, J., Vijayakumar, S., and Schaal, S. 2005. Natural Actor-Critic. In *European Conference on Machine Learning (ECML)*. Springer, 280–291.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. 257–286.

Raux, A., Langner, B., Bohus, D., Black, A. W., and Eskenazi, M. 2005. Lets go public! taking a spoken dialog system to the real world. In *Proc. of Interspeech 2005*.

Roy, N., Pineau, J., and Thrun, S. 2000. Spoken dialogue management using probabilistic reasoning. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, 93–100.

Schatzmann, J., Stuttle, M. N., Weilhammer, K., and Young, S. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding ASRU*.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., New York, NY, USA.

Sutton, R. and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass.

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

Syed, U. and Williams, J. D. 2008. Using automatically transcribed dialogs to learn user models in a spoken dialog system. In *HLT*. Morristown, USA, 121–124.

Thomson, B. 2010. Statistical methods for spoken dialogue management. Ph.D. thesis, University of Cambridge.

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

Wierstra, D., Foerster, A., Peters, J., and Schmidhuber, J. 2007. Solving deep memory POMDPs with recurrent policy gradients. In *In International Conference on Artificial Neural Networks*.

Wierstra, D., Frster, A., Peters, J., and Schmidhuber, J. 2010. Recurrent policy gradients. *Logic Journal of the IGPL 18,* 5, 620–634.

Williams, J. 2008a. Demonstration of a POMDP voice dialer. In *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*. Association for Computational Linguistics, Morristown, NJ, USA, 1–4.

Williams, J. D. 2008b. Integrating expert knowledge into POMDP optimization for spoken dialog systems. In *Proceedings of the AAAI-08 Workshop on Advancements in POMDP Solvers*.

Williams, J. D. and Young, S. 2005. Scaling Up POMDPs for Dialog Management: The Summary POMDP Method. In *IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*. Cancun, Mexico.

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

Young, S. 2007. CUED standard dialogue acts: http://mi.eng.cam.ac.uk/research/dialogue/LocalDocs/dastd.pdf. Tech. rep., Cambridge University Engineering Dept.

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

Zhang, B., Cai, Q., Mao, J., and Guo, B. 2001. Planning and Acting under Uncertainty: A New Model for Spoken Dialogue System. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*. Seattle.

Received July 2010; revised November 2010; accepted November 2010