Hyper-parameter Optimisation of Gaussian Process Reinforcement Learning for Statistical Dialogue Management

Lu Chen[1], **Pei-Hao Su**[2] and **Milica Gašić**[2*]
[1]Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
[2] Department of Engineering, University of Cambridge, Cambridge, UK
chenlusz@sjtu.edu.cn, phs26@cam.ac.uk, mg436@eng.cam.ac.uk

Abstract

Gaussian processes reinforcement learning provides an appealing framework for training the dialogue policy as it takes into account correlations of the objective function given different dialogue belief states, which can significantly speed up the learning. These correlations are modelled by the *kernel function* which may depend on *hyper-parameters*. So far, for real-world dialogue systems the hyper-parameters have been hand-tuned, relying on the designer to adjust the correlations, or simple non-parametrised kernel functions have been used instead. Here, we examine different kernel structures and show that it is possible to optimise the hyper-parameters from data yielding improved performance of the resulting dialogue policy. We confirm this in a real user trial.

1 Introduction

Spoken dialogue systems enable human-computer interaction via speech. The dialogue management component has two aims: to maintain the dialogue *state* based on the current spoken language understanding input and the conversation history, and choose a response according to its dialogue *policy*. To provide robustness to the input errors, a number of statistical approaches are proposed to track a distribution over all dialogue states at every dialogue turn, called the *belief state* (Young et al., 2013; Thomson and Young, 2010; Williams et al., 2013; Henderson et al., 2014; Sun et al., 2014). The system response is then based on the belief state, rather than an inaccurate estimate of the most likely dialogue state.

The state-of-art statistical methods for policy learning are based on reinforcement learning (RL) (Young et al., 2013), which makes it possible to learn from interaction with the users. However, most RL methods take too many dialogues for policy training. In Gaussian process reinforcement learning (GPRL) the *kernel function* defines prior correlations of the objective function given different belief states, which can significantly speeds up the policy optimisation (Gašić and Young, 2014). Alternative methods include Kalman temporal difference (KTD) reinforcement learning (Pietquin et al., 2011). Typically, statistical approaches to dialogue management rely on the belief state space compression into a form of a *summary space*, where the policy learning can be tractably performed (Williams and Young, 2007; Pinault et al., 2009; Thomson and Young, 2010; Crook and Lemon, 2011). GPRL allows the learning to be performed directly on the full belief state. However, only non-parametrised kernel functions have been considered for this purpose (Gašić and Young, 2014).

Here we address the important problem of how to define the structure of the kernel function for a real-world dialogue task and learn the hyper-parameters from data for a policy that operates on the full belief state. Using only a small-size dataset for hyper-parameter optimisation, we show that the policy with the optimised kernel function outperforms both the policy with hand specified kernel parameters and the one with a standard non-parametrised kernel function. This is particularly beneficial for policy training with real users.

This paper is organised as follows. In section 2, we briefly review GP-Sarsa and the hyper-parameter optimisation. Section 3 introduces the kernel functions examined here. The experimental results are shown in section 4, followed by conclusions and future work directions in section 5.

[*]Lu Chen was supported by the NICaiA project (the EU FP7 No. 247619). Pei-Hao Su is supported by Cambridge Trust and the Ministry of Education, Taiwan.

Proceedings of the SIGDIAL 2015 Conference, pages 407–411,
Prague, Czech Republic, 2-4 September 2015. ©2015 Association for Computational Linguistics

2 GP-Sarsa and hyper-parameter optimisation

The expected cumulative reward given belief state $\mathbf{b}$ and action a is defined by the Q-function as:

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

where r_τ is the immediate reward at τ-th dialogue turn, T is the number of dialogue turns and $\gamma \in [0, 1]$ is a discount factor. GP-Sarsa is an online RL algorithm that models the Q-function as a Gaussian process (Engel et al., 2005). It makes the learning tractable by utilising the kernel span sparcification algorithm and constructing a set of representative belief state and action called the *dictionary*. The computational complexity of GP-Sarsa is $O(Tm^2)$ where m is the size of the dictionary and T is the number of turns of all interactions.

In the case of Gaussian process regression, the kernel function parameters can be estimated by *evidence maximisation* in such a way that they capture the correlations that occur in the data (Rasmussen and Williams, 2005). This approach has been extended for the case of GP-Sarsa, however its benefits have so far only been shown for a toy dialogue problem (Gašić et al., 2010).

Using a data corpus of belief state-action pairs and rewards, the hyper-parameters can be found by minimising the negative log marginal likelihood via a conjugate gradient method (Rasmussen and Nickisch, 2010) to find the optimal hyper-parameters. The computational complexity of the gradient calculation is $O(nT^3)$, where n is the number of hyper-parameters and T is the total number of dialogue turns in the corpus (see appendix A).

3 Kernel functions

In Gaussian process regression, the kernel function $k(\cdot, \cdot)$ must be positive definite (Rasmussen and Williams, 2005). The kernel functions have some interesting properties (Duvenaud et al., 2013). If k_1 and k_2 are kernels,

- $k_1 + k_2$ is kernel. Adding two kernels can be thought of as an OR-like operation, as the resulting kernel will have high value if either of the two base kernels have a high value.
- $k_1 \cdot k_2$ is kernel. Multiplying two kernels can be thought of as an AND-like operation, as the function value is only expected to be similar to some other function value if both kernels have a high value.

3.1 Standard kernel functions

There are many valid kernel functions (Rasmussen and Williams, 2005). The following three ones are the basic kernels used in our experiments.

- Gaussian kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = p^2 \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{2l^2})$, where p and l are the hyper-parameters. If the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is more than the lengthscale l, the outputs are uncorrelated. The output variance p^2 determines the average distance of the function from its mean.
- Linear kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- δ-kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \delta(\mathbf{x}_i, \mathbf{x}_j)$, where the function values are correlated if and only if the inputs are the same.

3.2 Kernels for dialogue management

The kernel for two belief-action pairs is normally decomposed as the product of separate kernels over belief states and actions, $k_B(\cdot, \cdot)$ and $k_A(\cdot, \cdot)$.

The elements of the dialogue state are *concepts* that occur in the dialogue and are described by the underlying *ontology*. For instance, in a restaurant information domain, these usually include *goal-food*, *goal-area*, *history-food*, *history-area* and so on. The belief tracker maintains a probability distribution for each of them. We first define kernel function on each concept $k_{B_i}(\cdot, \cdot)$, then combine them to form the kernel function for the whole belief state $k_B(\cdot, \cdot)$, as illustrated in Figure 1.

The Gaussian kernel and the linear kernel were used as basic kernel functions for each concept in the belief state. In the case of Gaussian kernels, if they share the same hyper-parameters across different concepts, we refer to them as *concept independent*, otherwise, they are called *concept dependent*. While the linear kernel is used for problems involving distributions (Jebara et al., 2004), Gaussian kernel is a more natural choice here as the Q-function for belief states is a non-linear smooth function. Additionally, we investigated two integration methods: one is to sum up the kernels for all concepts, the *sum kernel*; another is to multiply kernels for all concepts, the *product kernel*.

The action kernel is defined on summary actions and given that the total number of summary actions is usually small, e.g. 20, the δ-kernel is chosen as the action kernel.

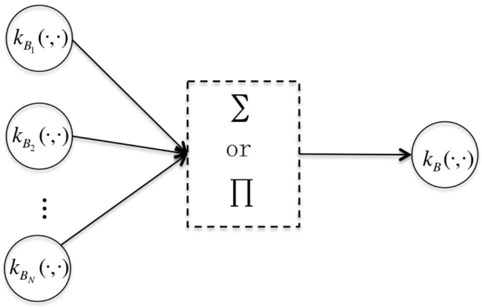


Figure 1: Kernel structure for belief state

4 Experiments and results

The training data for hyper-parameter optimisation was generated using an agenda-based user simulator (Schatzmann et al., 2007). The dialogue policy training and evaluation is performed both on the user simulator and human users. The system operates on the TopTable dialogue domain which consists of about 150 restaurants in Cambridge, UK (TopTable, 2012). Each restaurant has 9 slots, e.g. food, area, phone, address and so on. The state decomposes into 21 concepts. Each concept takes from 4 to 150 values. Each value is given a belief in $[0, 1]$ by the BUDS state tracker (Thomson and Young, 2010). The summary action space consists of 20 summary actions.

4.1 Experimental procedure

The hyper-parameter are optimised as follows.

1. **Training data generation**: We used a random policy to generate simulated dialogues out of which a small number of *successful* dialogues were used as training data.[1]
2. **Interval estimation**: We estimated appropriate *intervals* for concept independent hyper-parameters according to the properties of Gaussian kernel as described in section 3.1. In our experiments, we only restricted the range of lengthscale l. For the sum Gaussian kernel, the belief state for each concept is a probability distribution, so the lengthscale l is in interval $(0, \sqrt{2}]$. For product Gaussian kernel, the product of Gaussian kernels is still a Gaussian kernel, therefore the lengthscale l should be less than the maximum distance between two whole belief states (5.29 in the TopTable domain).
3. **Concept independent hyper-parameter optimisation**: We sampled initial concept independent hyper-parameters from the estimated intervals and then minimised the negative log likelihood to find the concept independent optimised hyper-parameters. We repeated this N times, and the hyper-parameters with the overall smallest negative log likelihood was chosen as the final concept independent hyper-parameters.
4. **Concept dependent hyper-parameter optimisation**: We initialised them as concept independent hyper-parameters, then minimised the negative log likelihood to get the concept dependent optimised hyper-parameters.

Name	Kernel	Learnt	Combine	Concept dep.
GHSI	Gaussian	N	Sum	N
GLSI	Gaussian	Y	Sum	N
GLSD	Gaussian	Y	Sum	Y
LS	Linear	/	Sum	/
GHPI	Gaussian	N	Prod	N
GLPI	Gaussian	Y	Prod	N
GLPD	Gaussian	Y	Prod	Y
LP	Linear	/	Prod	/

Table 1: Summary of kernels

After the hyper-parameters are obtained, we trained and evaluated the policies with these optimised kernels. For comparison, the policies with hand-tuned Gaussian kernel hyper-parameters and linear kernel were also trained and evaluated.

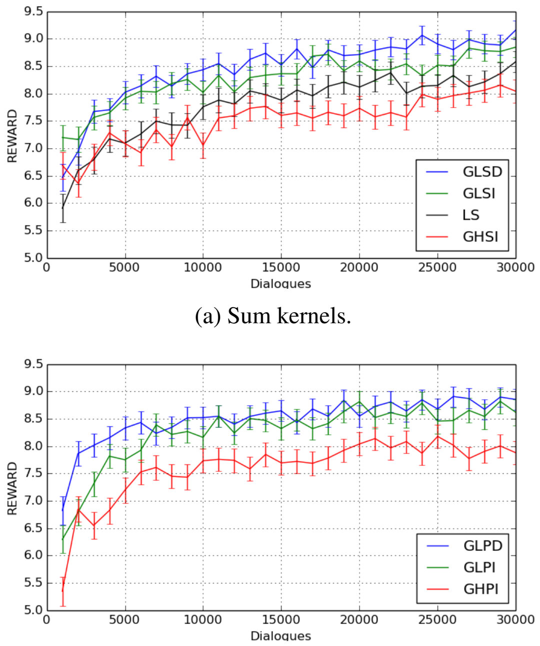
4.2 The results on the user simulator

During training, intermediate policies were recorded at every 1000 dialogues. Each policy was then evaluated using 1000 dialogues when testing. The reward was calculated as 20 for a successful dialogue, deducted for the number of turns.

We compared four different sum and product kernels (Table 1) and the results are given in Figure 2. The results in Figure 2(a) show that the policies with optimised sum Gaussian kernels perform significantly better than the policy using hand-tuned hyper-parameters (GHSI) and the linear kernel (LS). Also, in the later learning stages, the policy with concept dependent kernel (GLSD) appears to have reached a better performance than the one with concept independent kernel (GLSI).

The policies using the product kernels follow similar trends, except that the concept dependent product kernel (GLPD) performs significantly bet-

[1] We used 73 dialogues for concept independent and 147 dialogues for concept dependent hyper-parameter optimisation, with respectively 505 and 1004 dialogue turns. We found that the smaller data set was not sufficient to capture correlations for the concept dependent kernels.



(a) Sum kernels.

(b) Product kernels.

Figure 2: Comparison of policies with two kernels. Vertical bars denote standard errors. The average success rates for GHSI and GHPI are respectively 91.8% and 92.9% at the end of training.

ter than other kernels at the initial stages of training (Figure 2(b)).[2] The best performing product (GLPD) and sum (GLSD) kernel converge to similar performance, with the product kernel performing better in the early stages of training, at the expense of a larger dictionary.

4.3 Human experiments

In order to further evaluate the effect of the optimised kernels, policies were trained using crowdsourcing via the Amazon Mechanical Turk service in a set-up similar to (Jurčíček et al., 2011; Su et al., 2015). At the end of each dialogue, a recurrent neural network (RNN) model was used to predict the dialogue success used as the reinforcement feedback (Su et al., 2015).

The GLSD kernel and the GLPD kernel were selected for on-line policy training and compared to the LS kernel. Figure 3 shows the learning curve of the reward during training, demonstrating the advantage of Gaussian kernels over the simple linear kernel.

To confirm this result, all optimised policies were evaluated after 500 training dialogues. The results are given in Table 2. It can be seen that the policies with optimised kernels, especially the GLPD kernel, perform much better than the policy with linear kernel. [3]

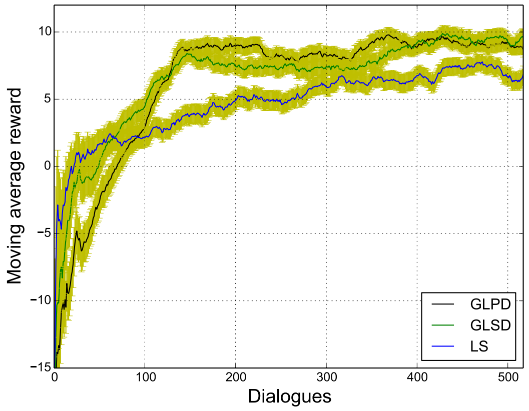


Figure 3: Learning curve of reward during on-line policy optimisation. For both plots, the moving average was calculated using a window of 100 dialogues. Yellow lines are standard errors.

Kernel	#Diags	Reward	Success(%)
LS	347	8.46 ± 0.57	77.2 ± 2.3
GLSD	336	9.56 ± 0.56	79.5 ± 2.2
GLPD	423	10.52 ± 0.47	82.3 ± 1.9

Table 2: Evaluation of policies with three kernels.

5 Conclusions and Future work

This paper has investigated the problem of kernel structure and hyper-parameter optimisation of Gaussian process reinforcement learning for statistical dialogue management. We have demonstrated that the optimised kernels yield significant improvements in the policy performance both when training with a simulated user and real users.

The work in this paper has focused on optimising the kernel function for the belief state space off-line. The future work will consider joint optimisation of the hyper-parameters and the policy. This will rely on finding a less computationally expensive method for hyper-parameter optimisation, also allowing more complex actions kernels to be investigated.

[2] The result of the product linear kernel (LP) is not reported due to poor performance. In the TopTable domain two belief states for *history* concepts are typically very different, so the linear kernel for these concepts is often close to 0. This results in a very small overall kernel value that leads to slow convergence.

[3] In (Gašić and Young, 2014), summary space-based policies were outperforming full-space policies because the summary space kernels could be regarded carefully hand-coded kernels on full-belief space and full-space kernels were not optimised.

References

Paul A Crook and Oliver Lemon. 2011. Lossless value directed compression of complex user goal states for statistical spoken dialogue systems. In *INTERSPEECH*, pages 1029–1032.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. 2013. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1166–1174.

Y Engel, S Mannor, and R Meir. 2005. Reinforcement learning with Gaussian processes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 201–208, New York, NY.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

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*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

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

Filip Jurčíček, Simon Keizer, Milica Gašić, Franois Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2011. Real user evaluation of spoken dialogue systems using amazon mechanical turk. In *Proceedings of Interspeech*, pages 3061–3064.

Olivier Pietquin, Matthieu Geist, and Senthilkumar Chandramohan. 2011. Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences. In *IJCAI 2011*, pages 1878–1883, Barcelona, Spain, July.

Florian Pinault, Fabrice Lefèvre, and Renato de Mori. 2009. Feature-based summary space for stochastic dialogue modeling with hierarchical semantic frames. In *INTERSPEECH*.

Carl Edward Rasmussen and Hannes Nickisch. 2010. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian processes for machine learning*. MIT Press.

J Schatzmann, B Thomson, K Weilhammer, H Ye, and SJ Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *HLT/NAACL*, Rochester, NY.

Pei-Hao Su, David Vandyke, Milica Gašić, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve Young. 2015. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. *Submitted to Interspeech.*

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014. A generalized rule based tracker for dialogue state tracking. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT).*

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

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

JD Williams and SJ Young. 2007. Scaling POMDPs for Spoken Dialog Management. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2116–2129.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL Conference*, pages 404–413.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

A Marginal log likelihood for GPRL

Algorithm 1 Log likelihood and gradient

```
Require:
    rewards $\mathbf{r}$, belief state-action pairs $\mathbf{B}$, Gram matrix: $\mathbf{K}(\boldsymbol{\theta})$, $\frac{\partial}{\partial\theta_j}(\mathbf{K}(\boldsymbol{\theta})+\sigma^2\mathbf{I})$, $\Theta=\{\boldsymbol{\theta},\sigma\}$, $\forall i, \mathbf{H}[i,i]=1, \mathbf{H}[i,i+1]=-\gamma, \mathbf{H}[i,j]=0, j\neq i, j\neq i+1$
Find $\boldsymbol{\alpha}$ so that $\mathbf{L}\mathbf{L}^\mathsf{T}\boldsymbol{\alpha}=\mathbf{r}$
$\mathcal{L}(\Theta)=\frac{1}{2}\mathbf{r}^\mathsf{T}\boldsymbol{\alpha}+\sum_{i=1}^{T}L_{ii}+\frac{T}{2}\log 2\pi$
Find $\mathbf{W}$ so that $\mathbf{L}\mathbf{L}^\mathsf{T}\mathbf{W}=\mathbf{I}$
For $j=0$ to $dim(\Theta)-1$ do
    $\mathbf{D}^j=\frac{\partial}{\partial\theta_j}(\mathbf{K}(\boldsymbol{\theta})+\sigma^2\mathbf{I})$
    $\frac{\partial}{\partial\theta_j}\mathcal{L}=-\frac{1}{2}\mathtt{tr}((\boldsymbol{\alpha}\boldsymbol{\alpha}^\mathsf{T}-\mathbf{W})\mathbf{H}\mathbf{D}^j\mathbf{H}^\mathsf{T})$
end for
return $\mathcal{L}(\Theta)$, $\frac{\partial}{\partial\theta_j}\mathcal{L}(\Theta)$
```