Published in the proceedings of the Workshop on "Structure & Priors in Reinforcement Learning" at ICLR 2019

META-LEARNING SURROGATE MODELS FOR SEQUENTIAL DECISION MAKING

Alexandre Galashov[*], Jonathan Schwarz[*],
Hyunjik Kim, Marta Garnelo, David Saxton, Pushmeet Kohli,
S.M. Ali Eslami[*2], Yee Whye Teh[*2]
DeepMind
London, United Kingdom
{schwarzjn,agalashov}@google.com

ABSTRACT

Meta-learning methods leverage past experience to learn data-driven inductive biases from related problems, increasing learning efficiency on new tasks. This ability renders them particularly suitable for sequential decision making with limited experience. Within this problem family, we argue for the use of such approaches in the study of model-based approaches to Bayesian Optimisation, contextual bandits and Reinforcement Learning. We approach the problem by learning distributions over functions using Neural Processes (NPs) (Garnelo et al., 2018b), a recently introduced probabilistic meta-learning method. This allows the treatment of model uncertainty to tackle the exploration/exploitation dilemma. We show that NPs are suitable for sequential decision making on a diverse set of domains, including adversarial task search, model-based reinforcement learning and recommender systems.

1 INTRODUCTION

Sequential decision making encompasses a large range of problems with many decades of research targeted at problems such as Bayesian optimisation, multi-armed contextual bandits or reinforcement learning. Recent years have brought great advances particularly in RL (e.g. Schulman et al., 2015; Silver et al., 2016), allowing the successful application of RL agents to increasingly complex domains. Yet, most modern machine learning algorithms require multiple magnitudes more experience than humans to solve even relatively simple problems. In this paper we argue that such systems ought to leverage past experience acquired by tackling related problem instances, allowing fast progress on a target task in the limited-data setting.

Consider the task of designing a motor controller for an array of robot arms in a large factory. The robots vary in age, size and proportions. The objective of the controller is to send motor commands to the robot arms in such a way that allows each one to achieve its designated task. The majority of current methods may tackle the control of each arm as a separate problem, despite similarity between the arms and tasks they may be assigned to. Instead, we argue that availability of data on related problems ought to be harnessed and discuss how learning *data-driven priors* can allow learning of a controller for additional robot arms in a fraction of the time.

A second question that immediately arises in the design of such a controller is how to deal with uncertainty: uncertainty about the proportions of each robot, uncertainty about the physics of their motor movements, and the state of the environment. We thus argue that a suitable method should allow reasoning about predictive uncertainty, rapidly adjusting these estimates once more data becomes available.

Much recent research has studied the problem of decision making under such uncertainty. Frameworks such as Bayesian Optimisation (e.g. Moćkus et al., 1978; Schonlau et al., 1998) and Contextual Bandits (e.g. Cesa-Bianchi & Lugosi, 2006) offer ways of efficiently balancing exploration and exploitation to simultaneously resolve uncertainties while making progress towards each robot arm's objective. This is often achieved by specifying some sort of *prior* belief model over the dynamics of the system in advance (e.g. a Gaussian Process), which is then used to balance exploration and exploitation. Importantly, note that significant domain-knowledge may be required to make an appropriate choice of a prior.

In this paper, we propose a strategy for combining the efficiency of traditional techniques for sequential decision making with the flexibility of learning-based approaches. We employ the framework of Neural Processes (NPs) (Garnelo et al.,

[*,*2]: Joint first/senior authorship

2018b) to instead *learn suitable priors* in a data-driven manner, and utilise them as part of the inner loop of established sequential decision making algorithms.

2 NEURAL PROCESSES (NPS)

Neural processes (NPs) are a family of models for few-shot learning (Garnelo et al., 2018b). Given a number of realisations from some unknown stochastic process $f : \mathcal{X} \to \mathcal{Y}$, NPs can be used to predict the values of f at some new, unobserved locations. In contrast to standard approaches to supervised learning such as linear regression or standard neural networks, NPs model a distribution over functions that agree with the observations provided so far (similar to e.g. Gaussian Processes (Rasmussen, 2003)). This is reflected in how NPs are trained: We require a dataset of evaluations of similar functions $f_1, \cdots f_n$ over the same spaces $\mathcal{X}$ and $\mathcal{Y}$. Note however, that we do not assume each function to be evaluated at the same $x \in \mathcal{X}$. Examples of such datasets could be the temperature profile over a day in different cities around the world or evaluations of functions generated from a Gaussian process with a fixed kernel. We provide further examples throughout the paper. For the equation governing the computational graph of a NP, please refer to the Appendix.

3 SEQUENTIAL DECISION MAKING WITH NP SURROGATE MODELS

We now discuss how to apply NPs to various instances of the sequential decision making problem. In all cases, we will make choices under uncertainty to optimise some notion of utility. A popular strategy for such problems involves fitting a surrogate model to observed data at hand, making predictions about the problem and constantly refining it when more information becomes available. Resulting predictions in unobserved areas of the input-space allow for planning and more principled techniques for exploration, making them useful when data efficiency is a particular concern.

3.1 BAYESIAN OPTIMISATION

We first consider the problem of optimising black-box functions without gradient information. A popular approach is Bayesian Optimisation (BO) (e.g. Shahriari et al., 2016), where we wish to find the minimiser $x^* = \arg\min_{x \in \mathcal{X}} f(x)$ of a (possibly noisy) function on some space $\mathcal{X}$ without requiring access to its derivatives. The BO approach consists of fitting a probabilistic surrogate model to approximate the function f on a small set of evaluations $\mathcal{C} = \{(x_i, y_i)\}$ observed thus far. Examples of a surrogate are Gaussian Processes, Tree-structured Parzen (density) estimators, Bayesian Neural Networks or for the purpose of this paper, NPs. The decisions involved in the process is the choice of some x' at which we choose to next evaluate the function f. This evaluation is typically assumed to be costly, e.g. when the optimisation of a machine learning algorithm is involved (Snoek et al., 2012). Typically, in addition to providing a good function approximation from limited data, we thus require a suitable model to provide good uncertainty estimates, which will be helpful to address the inherent exploration/exploitation trade-off during decision making.

In addition, we require an acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$ to guide decision making, designed such that we consider $x = \arg\max_{x'} \alpha(x')$ at the next point for evaluation. Model uncertainty is typically incorporated into α, as is done in some popular choices such as expected improvement (Moćkus et al., 1978) or the UCB algorithm (Srinivas et al., 2009). Given our surrogate model of choice for this paper, the Thompson sampling (Thompson, 1933; Chapelle & Li, 2011) criterion is particularly convenient (although others could be used). We show how this can be done with NPs in Algorithm 1.

3.2 MODEL-BASED REINFORCEMENT LEARNING

Next, we explain how to use NPs for reinforcement learning (RL) (Sutton & Barto, 2018). An RL problem is defined by (possibly stochastic) functions $f_t : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ (defining the transitions between states given an agent's actions) and the reward function $f_r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. These functions are together often referred to as an environment. We obtain the necessary distribution over functions for NP training by changing the properties of the environment, writing $p(\mathcal{T})$ to denote the distribution over functions for each task w, i.e. $\mathcal{T}_w = (f_t^w, f_r^w)$. The objective of the RL algorithm for a fixed task w is $\max_\theta \mathbb{E}_{\pi_\theta}[\sum_{t \geq 0} \gamma^t r_t^w]$, for rewards r^w obtained by acting on $\mathcal{T}_w$. $\pi_\theta : \mathcal{S} \to \mathcal{A}$ is a policy with parameters θ, i.e. the agent's decision making process. We also introduce $\gamma \in [0, 1]$, a discounting factor and t, indicating a time index in the current episode.

In this paper, we will focus our attention to a particular set of techniques referred to as model-based algorithms. Model-based RL methods assume the existence of some approximation $\hat{f}_t, \hat{f}_r$ to the dynamics of the problem at hand (typically learned online). Examples of this technique are (e.g. Peng & Williams, 1993; Browne et al., 2012).

We apply Neural Processes to this problem by first meta-learning an environment model using some exploratory policy π_ϕ (e.g. temporarily-correlated random exploration) on samples of the task distribution $p(\mathcal{T})$. This gives us an environment-model capable of quickly adapting to the dynamics of problem instances within the task distribution.

Thereafter, we use the model in conjunction with any RL algorithm to learn a policy π_ψ for a specific task $\mathcal{T}^* \sim p(\mathcal{T})$. This can be done by by autoregressively sampling rollouts from the NP (i.e. by acting according to π_ψ and sampling transitions using the NP's approximation to $\hat{f}_t^*, \hat{f}_r^*$. These rollouts are then used to update ψ using the chosen RL algorithm (sometimes referred to as indirect RL). Optionally, we may also update ψ using the real environment rollouts (direct RL). Algorithm 2 shows the proposed approach in more details.

Note that the linear complexity of an NP is particularly useful in this problem: As we allow for additional episodes on the real environment, the number of transitions that could be added to a context set grows quickly ($\mathcal{O}(mk)$ for m episodes of k steps). In complex environments, this may quickly because prohibitive for other methods (e.g. Gaussian Processes environment models (Deisenroth & Rasmussen, 2011)).

4 Problem setup

4.1 Adversarial Task-search for RL agents

As modern machine learning methods are approaching sufficient maturity to be applied in the real world, understanding failure cases of intelligent systems has become an important topic in our field, leading to the improvement of robustness and understanding of complex algorithms.

Inspired by recent work on adversarial attacks (e.g. Szegedy et al., 2013; Goodfellow et al., 2014; Madry et al., 2017; Behzadan & Munir, 2017; Huang et al., 2017; Uesato et al., 2018), we consider the recent study of (Ruderman et al., 2018) concerning failures of pretrained RL agents. The authors show that supposedly superhuman agents trained on simple navigation problems in 3D-mazes catastrophically fail when challenged with adversarially designed task instances trivially solvable by human players. The authors use an evolutionary search technique that modifies previous examples based on the agent's episode reward. A crucial limitation of this approach is that this technique produces out-of-distribution examples, weakening the significance of the results.

Instead, we propose to tackle the worst-case search through a Bayesian Optimisation approach on a *fixed set* of possible candidate maps using NP surrogate model. More formally, we study the adversarial task search problem on mazes as follows. For a given maze M, agent and goal positions p_a, p_g and obtained episode reward r, we consider functions of the form:

$$f_M : p_a, p_g \mapsto r \tag{1}$$

We can thus consider the following problems of interest:

(i) The worst-case position search for a fixed map: $p_a^*, p_g^* = \arg\min_{p_a, p_g} f_M(p_a, p_g)$

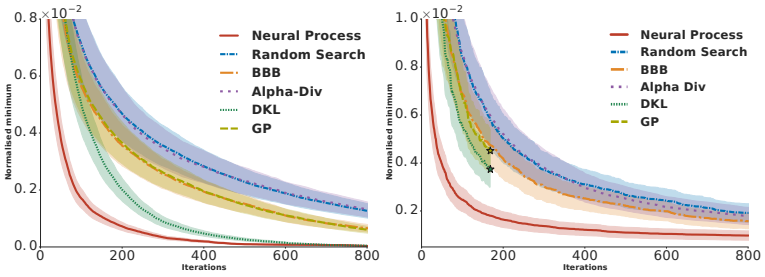
(ii) The worst-case search over a set of maps, including positions: $p_a^*, p_g^*, M^* = \arg\min_{p_a, p_g, M} f_M(p_a, p_g)$

We consider a fixed set of maps $\{M_1, \ldots, M_K\}$, such that for each map M, there is only a finite set (of capacity C) of possible agent and goal positions. For a fixed number of iterations N, the complexity of solving problem (ii) scales as $\mathcal{O}(NKC)$, where $\mathcal{O}(KC)$ corresponds to evaluation of the acquisition function α on all possible inputs. However, assuming a solution to (i) has already been found, we can reduce the complexity to $\mathcal{O}(\frac{N}{l}(K + lC))$, for some small l as follows: We use an additional *map model* $g : M \to \mathbb{R}$, which for a given map M directly predicts the minimum reward over all possible agent and goal positions, explaining the term $\mathcal{O}(K)$. Then, given the map M, we run our available solution to (i) for l iterations to find agent and goal positions. This corresponds to the term $\mathcal{O}(lC)$. We refer to this model as *position model*.

4.2 Model-based RL

As a classic RL problem, we consider the control task "cartpole" (Barto et al., 1983), defined fully by the physical parameters of the system. We obtain a distribution over tasks $p(\mathcal{T})$ (i.e. state transition and reward functions) by uniformly sampling the pole mass $p_m \sim \mathcal{U}[0.01, 1.0]$ and cart mass $c_m \sim \mathcal{U}[0.1, 3.0]$ for each episode, allowing pretraining of an NP. For the exploration policy π_ϕ required during pretraining, we use the following random walk:

$$a_t = sin(a_0 + u \sum_{k=1}^{t} w_k), \tag{2}$$



(a) Position model results (b) Full search problem results

Figure 1: Bayesian Optimisation results. **Left:** Position search **Right:** Full map search. We report the minimum up to iteration t (scaled in [0,1]) as a function of the number of iterations. Bold lines show the mean performance over 4 unseen agents on a set of held-out maps. We also show 20% of the standard deviation. Baselines: Gaussian Processes with a linear and Matern 3/2 product kernel (Bonilla et al., 2008), Bayes by Backprob (BBB) (Blundell et al., 2015), AlphaDivergence (AlphaDiv) (Hernández-Lobato et al., 2016), Deep Kernel Learning (DKL) (Wilson et al., 2016) and random search. In order to ensure a correct implementation we use the thoroughly tested code provided by the authors of Riquelme et al. (2018).

where $a_0 \sim \mathcal{U}[0, 2\pi]$, $u \sim \mathcal{U}[0, 1]$ are fixed for the entire episode and $w_k \sim \mathcal{N}(0, 1)$. Another option would be to use pretrained expert to give meaningful trajectory unrolls, especially for more complicated tasks. This however, requires a solution to at least one task within the distribution.

As the RL algorithm of choice, we use on-policy SVG(1) (Heess et al., 2015) without replay. We provide a comparison to the related model-free RL algorithm SVG(0) (Heess et al., 2015) with Retrace off-policy correction (Munos et al., 2016) as competitive baseline. For the experiments considered, we found that is was not necessary to update the policy using real environment trajectories.

5 Results and Analysis

5.1 Adversarial Task Search

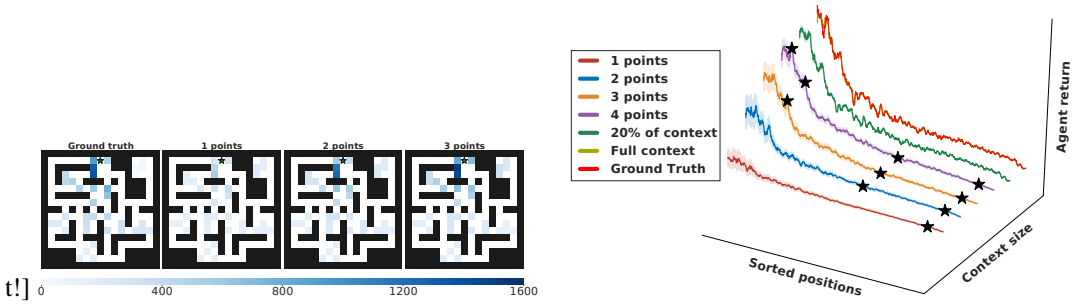
We consider a set of 1000 randomly chosen maps with 1620 agent and goal positions each, given a total agent population of 16 independently trained agents. We divide the maps into a 80% training and 20% holdout set. In order to encourage population diversity (both in terms of behaviour and performance), we trained each agent on both the task of interest, which is *explore_goal_locations_large* and four auxiliary tasks from DMLab-30 (Beattie et al., 2016), using a total of four different RL algorithms across the population. 12 agents (3 of each type) are used during pretraining, while we reserve the remaining 4 agents (1 of each type) for evaluation.

Addressing the question of performance of the *position model* first, we show results in Figure 1(a) indicating strong performance of our method. Indeed, we find agent and goal positions close to the minimum after evaluating only approx. 5% of the possible search space.

In order to explain the sources of this improvement, we show an analysis of NP uncertainty in function space in Figure 2(b) for varying context sizes. The graphic should be understood as the equivalent of Figure 3 in (Garnelo et al., 2018b) for the adversarial task search problem. More specifically, we plot functions of the form equation 1 drawn from a neural process by sampling $z \sim q(z|\mathcal{C})$ using varying context sizes. As we would expect, uncertainty in function space decreases significantly as additional context points are introduced.

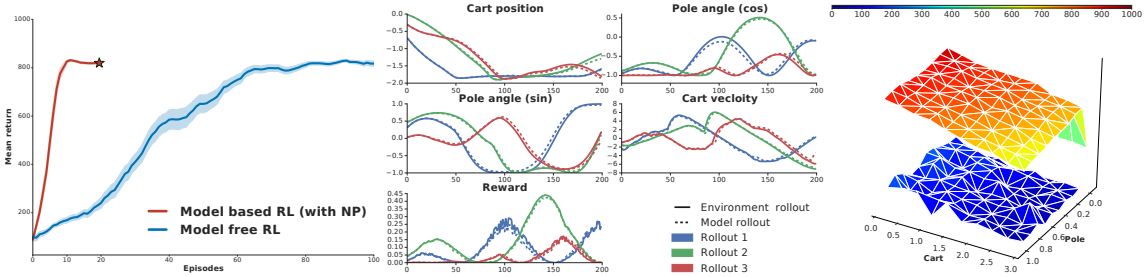
Showing the advantage of pretraining our method, we illustrate episode reward predictions on maps given small context sets in Figure 2(a). Note that the model has learned to assign higher scores to points closer to the starting location taking into account obstacles, without any explicitly defined distance metric provided. Thus, predictions incorporate this valuable prior information and are fairly close to the ground truth after a single observation. Indeed, subsequently added points appear to be mainly adjusting the scale of the predicted reward and the relative ordering of low-reward points.

Finally, we test our model in full case search on holdout maps, using the proposed two-stage approach to reduce search complexity as outlined above. From Figure 1(b), we continue to observe superior performance for this significantly more difficult problem.



(a) Few shot predictions (b) Uncertainty analysis

Figure 2: **Top:** Held-out agent episode returns for a fixed start (green star) and varying goal positions. Shown is the ground-truth and few-shot predictions by a NP. **Bottom:** Expected episode reward for an unseen agent over 1000 functions drawn from an NP. Positions are sorted by the absolute distance between agent and goal positions.



(a) Example learning curves. (b) Eenvironment and model rollouts using a random initial state and π_ψ . (c) Performance for different task parameters

Figure 3: **Left & Middle**: Learning curves (showing mean and standard deviation over 10 random repetitions.) and example rollouts for default task parameters $p_m = 0.1, c_m = 1.0$. **Right**: Mean episode reward at convergence for varying cart and pole masses.

5.2 Model-based RL

Results of the cartpole experiment are shown in Figure 3(a), We observe strong results, showing a model-based RL algorithm with an NP model can successfully learn a task in about 10-15 episodes. We show example videos for a particular run in [1]. Testing our method on the full support of the task distribution, we show the mean episode reward in Figure 3(c) (comparing to a random policy in blue). We observe that the same method generalises for all considered tasks. As expected, the reward decreases slightly for particularly heavy carts. We also provide a comparison of NP rollouts comparing to the real environment rollouts in Figure 3(b).

6 Discussion

In this paper, we have demonstrated the use of Neural Processes to learn *data-driven priors* for decision problems over a diverse set of problems, showing competitive results. As an additional experiment, we consider recommender systems as instances of the multi-armed bandit problem in the Appendix. At this point we would like to remind the reader that no aspect of the models used in the experiments has been tailored towards the problems we study.

Our experiments on adversarial task search indicate that such a system may for instance be used within an agent evaluation pipeline to test for exploits. An interesting avenue of future work could utilise the presented method to train more robust agents, using the NP to suggest problems the agent is currently unable to solve (cf. curriculum learning).

Our RL experiments showed significant improvements in terms of data efficiency when a large set of related tasks is available. In future work, it would be interesting to consider more complex problems, which may require a more sophisticated policy during pretraining.

[1] https://goo.gl/9yKav3

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Philip Bachman, Riashat Islam, Alessandro Sordoni, and Zafarali Ahmed. Vfunc: a deep generative model for functions. *arXiv preprint arXiv:1807.04106*, 2018.

Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, SMC-13(5):834–846, 1983.

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartłomiej Świątkowski, Bernhard Schölkopf, and Richard E Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich KÃijttler, Andrew Lefrancq, Simon Green, VÃ■ctor ValdÃl's, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. URL `https://arxiv.org/abs/1612.03801`.

Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275. Springer, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Edwin V. Bonilla, Kiang Ming A. Chai, and Christopher K.I. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2008.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pp. 2249–2257, 2011.

Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, 13(Dec):3619–3622, 2012.

Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 748–756. JMLR. org, 2017.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*, 2013.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model agnostic meta-learning. *arXiv prerint arXiv:1806.02817*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *ICML*, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018b.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Daniel Hernández-Lobato, Thang Bui, and Richard Turner. Black-box α-divergence minimization. In *International Conference on Machine Learning*, 2016.

Luke B Hewitt, Maxwell I Nye, Andreea Gane, Tommi Jaakkola, and Joshua B Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. *arXiv preprint arXiv:1807.08919*, 2018.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434. ACM, 2008.

Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pp. 2447–2455, 2011.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In *International Conference on Machine Learning*, pp. 82–90, 2013.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670. ACM, 2010.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

J Moćkus, V Tiesis, and A Źilinskas. The application of bayesian methods for seeking the extremum. vol. 2, 1978.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1046–1054, 2016.

Jing Peng and Ronald J Williams. Efficient learning and planning within the dyna framework. *Adaptive Behavior*, 1(4): 437–454, 1993.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2019.

Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*, 2017.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Avraham Ruderman, Richard Everett, Bristy Sikder, Hubert Soyer, Jonathan Uesato, Ananya Kumar, Charlie Beattie, and Pushmeet Kohli. Uncovering surprising behaviors in reinforcement learning via worst-case analysis. 2018. URL `https://openreview.net/forum?id=SkgZNnR5tX`.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pp. 880–887. ACM, 2008.

Matthias Schonlau, William J Welch, and Donald R Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, pp. 11–25, 1998.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 111–112. ACM, 2015.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Florian Strub, Romaric Gaudel, and Jérémie Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 11–16. ACM, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *Proceedings of the 28th International Conference on machine learning (ICML)*, 2018.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management*, pp. 337–348. Springer, 2008.

7 APPENDIX

7.1 NEURAL PROCESSES

To mimic the desired behaviour during training we split evaluations (x_i, y_i) for each function f into two disjoint subsets: a set of m context points $\mathcal{C} = \{(x_i, y_i)\}_{i=1}^{m}$ and a set of targets $\mathcal{T} = \{(x_j, y_j)\}_{j=m+1}^{n}$ that contains $n - m$ unobserved points. These data points are then processed by a neural network as follows:

$$r_i = h_\theta(x_i, y_i) \qquad \forall (x_i, y_i) \in \mathcal{C} \tag{3}$$

$$r = r_1 \oplus r_2 \oplus \dots r_{n-1} \oplus r_n \tag{4}$$

$$z \sim \mathcal{N}(\mu_{\psi_1}(r, \log(n)), \sigma_{\psi_2}(r, \log(n))) \tag{5}$$

$$\phi_i = g_\omega(x_j, z) \qquad \forall (x_j) \in \mathcal{T} \tag{6}$$

First, we use an encoder h_θ with parameters, transforming all (x_i, y_i) in the context set to obtain representations r_i. We then aggregate all r_i to a single representation r using a permutation invariant operator $\oplus$ (such as addition) that captures the information about the underlying function provided by the context points. Later on, we parameterise a distribution over a latent variable z, here assumed to be Normal with μ, σ estimated by an encoder network using parameters ψ_1, ψ_2. Note that this latent variable is introduced to model uncertainty in function space, extending the Conditional Neural Process (Garnelo et al., 2018a).

Thereafter, a decoder g_ω is used to obtain predictive distributions at target positions $x_i \in \mathcal{T}$. Specifically, we have $p(y_i|x_i, z; \phi_i)$ with parameters ϕ_i depending on the data modelled. In practice, we might decide to share parameters, e.g. by setting $\theta \subset \omega$ or $\psi_1 \cap \psi_2 \neq \varnothing$. To reduce notational clutter, we suppress dependencies on parameters from now on.

In order to learn the resulting intractable objective, approximate inference techniques such as variational inference are used, leading to the following evidence lower-bound:

$$\begin{aligned} &\log p(y_{m+1:n}|x_{1:n}, y_{1:m}) \\ \geq &\mathbb{E}_{q(z|x_{1:n}, y_{1:n})}\left[\sum_{i=m+1}^{n} \log p(y_i|z, x_i) + \log \frac{p(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})}\right] \end{aligned} \tag{7}$$

which is optimised with mini-batch stochastic gradient descent using a different function f_j for each element in the batch and sampling $|\mathcal{C}|, |\mathcal{T}|$ at each iteration.

7.2 RELATED WORK

There has been a recent surge of interest in Meta-Learning or Learning to learn, resulting in large array of methods (e.g. Koch et al., 2015; Andrychowicz et al., 2016; Wang et al., 2016; Reed et al., 2017), many of which may be applied in the problems we study (as we merely assume the existence of a general method for regression). However, predictive uncertainty may not be available for the majority of methods.

However, several recent publications focus on probabilistic ideas or re-interpretations of popular methods (e.g. Bauer et al., 2017; Rusu et al., 2018; Bachman et al., 2018), and could thus be suitable for the problems we study. An example is Probabilistic MAML (Finn et al., 2018) which forms an extension if the popular Model-agnostic meta-learning algorithm (Finn et al., 2017) that can be learned with variational inference. Other recent works cast meta-learning as hierarchical Bayesian inference (e.g. Edwards & Storkey, 2016; Hewitt et al., 2018; Grant et al., 2018; Ravi & Beatson, 2019).

Gaussian Processes (GPs) are popular candidates due to closed-form Bayesian inference and have been used for several of the problems we study (e.g. Rasmussen, 2003; Krause & Ong, 2011; Deisenroth & Rasmussen, 2011). While providing excellent uncertainty estimates, the scale of modern datasets can make their application difficult, thus often requiring approximations (e.g. Titsias, 2009). Furthermore, their performance strongly depends on the choice of the most suitable kernel (and thus prior over function), which may in practice require careful design or compositional kernel search (e.g. Duvenaud et al., 2013).

Deep Kernel Learning (Wilson et al., 2016) provides an alternative, addressing scalability concerns while allowing the advantage of structural properties of deep learning architectures. The network weights are learned by considering them part of the kernel-hyperparameters, jointly optimised through the log-marginal likelihood.

Moreover, much of the recent work on Bayesian Neural Networks (e.g. Blundell et al., 2015; Gal & Ghahramani, 2016; Hernández-Lobato et al., 2016; Louizos & Welling, 2017) serves as a reasonable alternative, also benefiting from the flexibility and power of modern deep learning architectures.

Finally, the approach in (Chen et al., 2017) tackles similar problems, applying meta-learning for black-box optimization or BO. It skips the uncertainty estimation and is trained to directly suggest the next point for evaluation.

7.3 BAYESIAN OPTIMISATION

Algorithm 1 Bayesian Optimisation with NPs and Thompson sampling.

Input:
f - Function to evaluate
$\mathcal{C}_0 = \{(x_0, y_0)\}$ - Initial randomly drawn context set
N - Maximum number of function iterations
$\mathcal{NP}$ - Neural process pretrained on evaluations of similar functions $f_1, \dots f_n$

for n=1, …, N **do**
Infer conditional $\mathcal{NP}$ prior $q(z|C_{n-1})$
Thompson sampling: Draw $z_n \sim q(z|C_{n-1})$, find

$$x_n = \arg\min_{x \in \mathcal{X}} \mathbb{E}\big[g(y|x, z_n)\big] \tag{8}$$

Evaluate target function and add result to context set
$\mathcal{C}_n \leftarrow \mathcal{C}_{n-1} \cup \{(x_n, f(x_n))\}$
end for

7.4 CONTEXTUAL MULTI-ARMED BANDITS

7.4.1 PROBLEM SETUP

Closely related to Bayesian Optimisation, the decision problem known as a contextual multi-armed bandit is formulated as follows. At each trial t:

1. Some informative context x_t is revealed. This could for instance be features describing a user of an online content provider (e.g. Li et al., 2010). Crucially, we assume x_t to be independent of past trials.
2. Next, we are to choose one of k arms $a^1, \dots, a^k \in \mathcal{A}$ and receive some reward $r_t \sim p_{a_t}$ given our choice a_t at the current iteration t. The current context x_t, past actions and rewards $(x_\tau, r_\tau)_{\tau=1}^{t-1}$ are available to guide this choice. As $\mu_k = \mathbb{E}_{r \sim p_{a_k}}[r]$ is unknown, we face the same exploration/exploitation trade-off as in the BO case.
3. The arm-selection strategy is updated given access to the newly acquired (x_t, r_t). Importantly, no reward is provided for any of the arms $a \neq a_t$.

Neural Processes can be relatively straight-forwardly applied to this problem. Decision making proceeds as in Algorithm 1 (Thompson sampling being a reasonable choice) with the main difference being that we evaluate the NP separately for each arm a_i. Past data is easily incorporated in the context set by providing a one-hot vector indicating which arm a_t has been chosen, along with x_t and r_t.

However, in practice it might be significantly more difficult to find or design related bandit problems that are available for pretraining. We will discuss a natural real-world contextual bandit application suitable for NP pretraining in the experimental section.

7.5 RECOMMENDER SYSTEMS

Considering next the contextual multi-armed bandit problem discussed in Section 7.4, we apply NPs to recommender systems. Decisions made in this context are recommendations to a user. As we aim to learn more about a user's preferences, we can think about certain recommendations as more exploratory in case they happen to be dissimilar to

previously rated items. Indeed, the problem has previously been modelled as a contextual multi-armed bandit, e.g. by (Li et al., 2010), using linear models.

The application of an NP to this problem is natural: We can think of each user u as a function from items $\mathcal{I}$ to ratings $\mathcal{R}$, i.e. $f_u : \mathcal{I} \to \mathcal{R}$, where each user f_u is possibly evaluated on a different subset of items. Thus most available datasets for recommender systems naturally fit the requirements for NP pretraining. Connecting this to the general formulation of a contextual bandit problem, we can think of each arm a_k as a particular item recommendation and consider the user id and/or any additional information that may be provided as context x. Rewards $r \sim p_{a_t}$ are likely to be domain-specific, but may be as simple as the rating given by a user.

The sequential decision process in this case can be explicitly treated by finding a suitable acquisition function for recommender systems. While this choice most likely depends on the goals of a particular business, we will provide a proof-of-concept analysis, explicitly maximising coverage over the input space to provide a function approximation as well as possible. This is motivated by the root-mean-squared-error metric used in the literature on the experiments we will consider. Inspired by work on decision trees, a natural criterion for evaluation could be the information gain at a particular candidate item/arm.

Writing $\mathbf{r}_{\setminus i}$ to denote the reward for each arm except i in the target set $\mathcal{T}$ (likewise for $\mathbf{a}_{\setminus i}$) and suppressing dependence on latents z and the context x for clarity, we can thus define the information gain $\mathcal{IG}$ at arm a_i:

$$\begin{aligned}\mathcal{IG}(a_i) := \mathcal{H}\big(p(\mathbf{r}_{\setminus i}|\mathbf{a}_{\setminus i}, \mathcal{C})\big) - \\ \mathbb{E}_{\hat{r}_i \sim p(\mathbf{r}_i|a_i,\mathcal{C})}\big[\mathcal{H}\big(p(\mathbf{r}_{\setminus i}|\mathbf{a}_{\setminus i}, \mathcal{C} \cup \{a_i, \hat{r}_i\})\big)\big]\end{aligned} \tag{9}$$

Note that this involves using samples of the model's predictive distribution $\hat{r}_j$ at arm a_j to estimate the entropy given an additional item of information. Assuming $p(r_i|a_i, \mathcal{C})$ is a univariate normal, we arrive at an intuitive equation to determine the expected optimal next arm a^* for evaluation:

$$\begin{aligned}a^* &= \arg\max_{a_i} \mathcal{IG}(a_i) \\ &= \arg\min_{a_i} \mathbb{E}\big[\mathcal{H}\big(p(\mathbf{r}_{\setminus i}|\mathbf{a}_{\setminus i}, \mathcal{C} \cup \{a_i, \hat{r}_i\})\big)\big] \\ &= \arg\min_{a_i} \mathbb{E}\big[\frac{1}{2}\ln(|\Sigma|) + \frac{|\mathcal{T}|-1}{2}(1 + \ln(2\pi))\big] \\ &= \arg\min_{a_i} \mathbb{E}\big[\ln(\prod_i \sigma_i^2)\big]\end{aligned} \tag{10}$$

where we made use of conditional independence, the analytic form of the entropy of a multivariate normal and the determinant of a diagonal matrix. We thus seek to recommend the next item such that the product of variances of all other items in the target set given the user's expected response is minimised. At this point it is important to mention that the successful application of this idea depends on the quality of the uncertainty estimates used. This is a strength of the NP family.

7.5.1 RECOMMENDER SYSTEMS

We apply NPs to the Movielens 100k & 20m datasets (Harper & Konstan, 2016). While the specific format of the datasets vary slightly, in both cases we face the basic problem of recommending movies to a user, given side-information such as movie genre, and tags (20m only) or certain user features (100k only) such as occupation, age and sex. Importantly, while discrete ratings warrant treatment as an ordinal regression problem, we leave this for future work to allow our results to be directly comparable to (Chen et al., 2018).

Discussing first the smaller MovieLens 100k dataset, we closely follow the suggested experimental setup in (Chen et al., 2018). Importantly, 20% of the users are explicitly withheld from the training dataset to test for few-shot adaptation. This is non-standard comparing to mainstream literature, which typically use a fraction of ratings for known users as a test set. This is particular interesting for NPs, recalling their application at test time without gradient steps (see discussion in section 2). Provided this works to a satisfactory degree, this property may be particularly desirable for fast on-device recommendation on mobile devices. Finally, similar to the argument made in (Chen et al., 2018), NPs can be trained in a federated learning setting, which may be an important advantage of our method in case data privacy happens to be a concern.

Table 1: Results on MovieLens 100k (**left**) and 20m (**right**). For the 100k dataset, we report results on varying number of unseen ratings on new users. On MovieLens 20m we report results on 10% unseen ratings of *known* users. In both cases, we report the RMSE. Baseline results taken from (Chen et al., 2018) and (Strub et al., 2016).

Model	**20%** of user data	**50%**	**80%**
SVD++ (Koren, 2008)	1.0517	1.0217	1.0124
Baseline Neural Network	0.9831	0.9679	0.9507
MAML (Finn et al., 2017)	0.9593	0.9441	0.9295
NP (random)	0.9381	0.9148	0.9050
NP (info. gain)	**0.9370**	**0.8751**	**0.8060**

Model	**90%**
BPMF (Salakhutdinov & Mnih, 2008)	0.8123
SVDFeature (Chen et al., 2012)	0.7852
LLORMA (Lee et al., 2013)	0.7843
ALS-WR (Zhou et al., 2008)	0.7746
I-Autorec (Sedhain et al., 2015)	0.7742
U-CFN (Strub et al., 2016)	0.7856
I-CFN (Strub et al., 2016)	**0.7663**
NP (random)	0.7957

Algorithm 2 Model-based Reinforcement Learning with NPs

Training Input:
$\mathcal{NP}_\theta$ - Neural process with parameters θ to estimate $f_t : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $f_r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$
$p(\mathcal{T})$ - Task Distribution
π_ϕ - Exploratory policy.

while Pretraining not finished **do**
  Sample a task $\mathcal{T}_i \sim p(\mathcal{T})$.
  Obtain transitions $\mathcal{D} = \{(s \in \mathcal{S}, a \in \mathcal{A}, r \in \mathbb{R}, s' \in \mathcal{S})\}$ from $\mathcal{T}_i$ using π_ϕ.
  Randomly split $\mathcal{D}$ into context and target sets $\mathcal{C}, \mathcal{T}$.
  Take a gradient step to minimise θ wrt. $equation\ 7$
end while

Evaluation Input:
ψ - Policy parameters for target task $\mathcal{T}^*$, $\mathcal{R}$ - Replay
K - Rollout length

while true **do**
  Generate trajectory $\tau = (s_1, a_1, r_1, \ldots, s_K)$ on target task $\mathcal{T}^*$ (i.e. the real environment) using π_ψ.
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\tau\}$

  for n=1,..., N **do**
    Sample state $s_1 \sim \mathcal{R}$ observed on $\mathcal{T}^*$ to initialise a trajectory .
    Generate trajectory $\tau' = (s_1, a_1 \ldots, s_K)$ using π_ψ and autoregressive sampling from $\mathcal{NP}$ (given $\mathcal{R}$ as context).

    Update policy π_ψ using any RL algorithm.
  end for
end while

Results for random context sets of 20%/50%/80% of test users' ratings (as suggested by the authors) are shown in Table 1 (left). While these results are encouraging, the treatment as a decision making process using our acquisition function (denoted info. gain) leads to much stronger improvements.

For completeness, we also provide results on the much larger MovieLens 20m dataset, for which more competitive baselines are available. Unfortunately, we are unable to show results using our acquisition function, due to a lack of baselines and thus only evaluate our method with a random context set. Nevertheless, Table 1 (right) shows comparable results to state-of-the-art recommendation systems, despite this limitation. We also discuss several suggestions for improvements in the conclusion.

7.6 Model-based Reinforcement Learning

We show how NPs can be applied to model-based RL in Algorithm 2.