Probing Dynamic Environments with Informed Policy Regularization

Pierre-Alexandre Kamienny [1] **Matteo Pirotta** [1] **Alessandro Lazaric** [1] **Thibault Lavril** [1] **Nicolas Usunier** [1]
Ludovic Denoyer [1]

Abstract

We study the problem of learning exploration-exploitation strategies that effectively adapt to dynamic environments, where the task may change over time. While RNN-based policies could represent such strategies, in practice their training time is prohibitive and the learning process often converges to poor solutions. In this paper, we consider the case where the agent has access to a description of the task (e.g., a task id or task parameters) at training time. We propose a novel algorithm that regularizes the training of an RNN-based policy using informed policies trained to maximize the reward in each task, dramatically improving the sample efficiency of RNN-based policies, without losing their representational power. We evaluate our algorithm in a variety of environments where tasks may vary within each episode.

1. Introduction

An intelligent agent is said to fully master a *problem* when it is able to generalize from a few instances (tasks) and achieve the objective of the problem under many variations of the environment. For instance, children *know* how to ride a bike (i.e., the problem) when they can reach their destination irrespective of the specific bike they are riding, which requires to adapt to the weight of the bike, the friction of the brakes and tires, and road conditions (i.e., the tasks).

How to enable agents to generalize across tasks has been studied in *Multi-task Reinforcement Learning* (e.g., Wilson et al., 2007; Teh et al., 2017), *Transfer Learning* (e.g., Taylor & Stone, 2011; Lazaric, 2012) and *Meta-Reinforcement Learning* (Finn et al., 2017; Hausman et al., 2018; Rakelly et al., 2019). These works fall into two categories. *Learning to learn* approaches aim at speeding up learning on new tasks, by pre-training feature extractors or learning good initializations of policy weights (Raghu et al., 2019).

The online adaptation setting is a special case of a partially observed markov decision problem, where the unobserved variables are the descriptors of the current task. It is thus possible to rely on recurrent neural networks (RNNs) (Bakker, 2001; Heess et al., 2015), since they can theoretically represent optimal policies in POMDPs if given enough capacity. Unfortunately, the training of RNN policies has often prohibitive sample complexity and it may converge to suboptimal local minima.

To overcome this drawback, efficient online adaptation methods leverage the knowledge of the task at training time. The main approach is to pair an exploration strategy with the training of *informed policies*, i.e., policies taking the description of the current task as input. *Probe-then-Exploit* (PTE) algorithms (e.g., Zhou et al., 2019) operate in two stages. They first rely on an exploration policy to identify the task. Then, they commit to the identified task by playing the associated informed policy. *Thompson Sampling* (TS) approaches (Thompson, 1933; Osband et al., 2016; 2019) maintain a distribution over plausible tasks and play the informed policy of a task sampled from the posterior following a predefined schedule. PTE and TS are expected to be sample-efficient relatively to RNNs as learning informed policies is a fully observable problem. However, PTE and TS cannot represent effective exploration/exploitation policies in many environments. First, PTE exploration is not reward-driven thus wasting valuable time to estimate unnecessary information, Second, the switch between probing and exploiting is hard to tune and problem-dependent. TS cannot represent complex probing policies because it is constrained to executing informed policies, and cannot adopt a behavior that is not one of the possible informed policies. Another drawback of TS approaches is that the re-sampling frequency needs to be carefully tuned. These drawbacks make their application to *non-stationary environments*, where the task changes within an episode, even worse.

Recently, Humplik et al. (2019) proposed an alternative approach, *Task Inference* (TI), which trains a full RNN policy with the current task prediction as an auxiliary loss. TI avoids the suboptimality of PTE/TS by not constraining the

[1]Facebook AI Research, Paris, France. Correspondence to: Ludovic Denoyer <denoyer@fb.com>.

structure of the exploration/exploitation policy. However, in TI, the task descriptors are used as targets and not as inputs, so TI does not leverage the faster learning of informed policies. Moreover, the behavior of TI in non-stationary environments has not been studied.

We study in this paper the *online adaptation* setting where a single policy is trained for a fixed family of tasks. When facing a new task, the policy must then balance *probing*, to reduce the uncertainty about the current task, and *exploitation* to maximize the cumulative reward of the task.

We propose a new model, IMPORT (*InforMed POlicy RegularizaTion*), that combines the rich expressivity of RNNs with the efficient learning of informed policies. Our method thus allows to discover a policy able to probe any new environment, and to use the acquired information to solve a particular task. In comparison to existing methods, IMPORT is able to discover complex probe/exploitation trade-offs and to learn from less interactions with the environment, using less prior assumptions.

We now describe the setting, the architecture of the model, the underlying loss function and propose a set of experiments and performance on non-stationary environments.

2. Setting

Let M be the space of possible tasks. Each 2 M is associated to an episodic -MDP M $= (S; A; p\ ; r\ ;\ )$ whose dynamics p and rewards r are task dependent, while state and action spaces are shared across tasks and is the discount factor. The descriptor can be a simple id (2 N) or a set of parameters (2 R^d).

When the reward function and the transition probabilities are unknown, RL agents need to devise a strategy that balances exploration to gather information about the system and exploitation to maximize the cumulative reward. Such a strategy can be defined as the solution of a partially observable MDP (POMDP), where the hidden variable is the descriptor of the MDP. Given a trajectory $_t$ $= (s_1; a_1; r_1; : : : ; s_t\ _1; a_t\ _1; r_t\ _1; s_t)$, a POMDP policy $(a_t j\ _t)$ maps the trajectory to actions. In particular, the optimal policy in a POMDP is a history-dependent policy that uses $_t$ to construct a belief state b_t, which describes the uncertainty about the task at hand, and then maps it to the action that maximizes the expected sum of rewards (e.g., Kaelbling et al., 1998).

At training time, we assume the agent has unrestricted access to the descriptor of the tasks it interacts with. In particular, we consider the challenging case of dynamic (i.e., non-stationary) environments, where the task may change over time according to a fixed or random schedule, $_t$ being the value of at time t. Let $q(\ j\ _t; f\ _i g_{i=1}^{t\ 1})$ to be a history-dependent task distribution, then at each step t a new task $_t$ is drawn from q.[1] Leveraging the information gathered at training time, we expect the agent to learn an exploration strategy that is better suited for tasks in M and q. More formally, after T steps of training, the agent returns a policy $(a_t j\ _t)$ that is evaluated according to its average performance across tasks in M generated from q, i.e., $\mathsf{E}[\sum_{t=1}^{j\ j}\ ^{t\ 1} r_t\ ^t j\ ]$: where the expectation is taken over trajectories of full episodes , and $j\ j$ is the length of episode .

The objective is then to find an architecture for that is able to express strategies that perform the best w.r.t. the reward and, at the same time, can be efficiently learned even for moderately short training phases.

3. Method

We introduce IMPORT (*InforMed POlicy RegularizaTion*), a novel policy architecture (described in Fig. 1) for efficient online adaptation that combines the rich expressivity of RNNs with the efficient learning of informed policies. At train time, a shared policy head receives as input the current observation, together with either a (learned) embedding of the current task, or the hidden state of an RNN such that the informed policy and the RNN policy are learned simultaneously. At test time, the hidden state of the RNN replaces the task embedding, and the agent acts without having access to the current task.

Our approach leverages the knowledge of the task descriptor and informed policies to construct a latent representation of the task that is *purely reward driven*. Since is unknown at testing time, we use this informed representation to train a predictor based on a recurrent neural network. To leverage the efficiency of informed policies even in this phase, we propose an architecture *sharing parameters* between the informed policy and the final policy such that the final policy will benefit from parameters learned with privileged information. The idea is to constrain the final policy to stay close to the informed policy while allowing it to perform probing actions when needed to effectively reduce the uncertainty about the task. We call this approach InforMed POlicy RegularizaTion (IMPORT).

Formally, we denote by $(a_t j s_t;\ )$ and $_H(a_t j\ _t)$ the informed policy and the history-dependent policy that is used at test time. The informed policy $=\ f$ is the functional composition of f and , where $f\ : M\ !\ Z$ projects in a latent space Z R^k and $: S\ \ Z\ !\ A$ selects the action based on the latent representation. The idea is that $f\ (\ )$ captures the relevant information con-

[1] Notice that the definition of q is rich enough so that it can represent cases such as stationary, piece-wise stationary, and shifts with limited deviations.

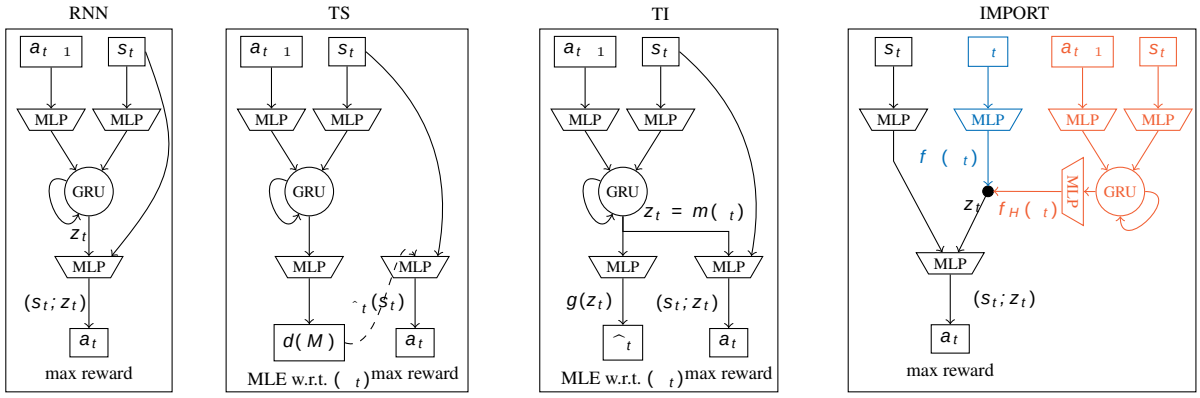

Figure 1. Representation of the different architectures. IMPORT is composed of two models sharing parameters: The (black+blue) architecture is the informed policy π optimized through **(B)** while the (black+red) architecture is the history-based policy π_H (used at test time) trained through **(A)+(C)**.

tained in while ignoring dimensions that are not relevant for learning the optimal policy. This behavior is obtained by training directly to maximize the task reward r .

While leverages the knowledge of at training time, $_H$ acts based on the sole history. To encourage $_H$ to behave like the informed policy while preserving the ability to probe, $_H$ and share , the mapping from latent representations to actions. We thus define as $_H$ = f_H where $f_H : H \; ! \; Z$ encodes the history into the latent space. By sharing the policy head , the approximate belief state constructed by the RNN is mapped to the same latent space as . When the uncertainty about the task is small, $_H$ then benefits from the joint training with .

More precisely, let $; !;$ the parameters of $; f_H$ and f respectively, so that $(a_t j s_t; \; _t)$ = f = $(a_t j s_t; f \; (\; _t))$ and $^{!}_{H} (a_t j \; _t)$ = $f^{!}_{H}$ = $(a_t j s_t; f^{!}_{H}(\; _t))$. The goal of IMPORT is to maximize over $; !;$ the objective function of Equation 1 with = 0.

$$\underbrace{\mathbb{E}\left[\sum_{t=1} \; _t r_t \; ^t \; \; ^{!};_H\right]}_{\textbf{(A)}} + \underbrace{\mathbb{E}\left[\sum_{t=1} \; _t r_t \; ^t \; \; ;\right]}_{\textbf{(B)}} + \underbrace{\mathbb{E}\left[\sum_{t=1} D \; f \; (\; _t); f_H(\; _t) \; \; ^{!};_H\right]}_{\textbf{(C)}} \quad (1)$$

Speeding Up the Learning. The optimization of **(B)** in Eq. 1 produces a reward-driven latent representation of the task through f . In order to encourage the history-based policy to predict a task embedding close to the one predicted by the informed policy, we augment the objective with an auxiliary loss **(C)** weighted by > 0. D is the squared 2-norm in our experiments. Note that the objective **(C)** is an auxiliary loss, so only the average gradient of D with respect to f_H along trajectories collected by $_H$ is backpropagated, ignoring the effect of f_H on $_H$.

Figure 2. Cartpole with task identifiers, R = 100 and ρ = 0.1.

4. Experiments

We compare our model to the following baselines: **Recurrent Neural Networks (RNN)** is a recurrent policy based using GRUs (Heess et al., 2015) that never uses . **Thompson Sampling (TS)** and **Task Inference (TI)** models are trained in the same setting as **IMPORT**, using at train time, but not at test-time. Each approach is trained using A2C (Mnih et al., 2016). Precise values of the considered hyper-parameters are given in Appendix A.2. The networks for all approaches have a similar structure with the same number of hidden layers and units for fair comparison. In all the tables, we report the test performance averaged over 100 test tasks (standard deviation over 3 seeds) of the best hyper-parameter values selected on 100 validation tasks (see Appendix A.1 for more details).

We perform experiments on a set of environments whose goals are to showcase different aspects: a) generalization to unseen tasks, b) adaptation to non-stationarities, c) ability to learn complex probing policies and d) sensitivity to the

Method	Test reward					
	$_{max} = 0.9$ ($_{min} = 0.1$)			$_{max} = 0.5$ ($_{min} = 0.1$)		
	$K = 5$	$K = 10$	$K = 20$	$K = 5$	$K = 10$	$K = 20$
UCB	64.67(0.38)	52.03(1.13)	37.61(1.5)	31.12(0.21)	22.84(0.97)	**15.63(0.74)**
SW-UCB	62.17(0.74)	50.8(1.21)	33.02(0.19)	30.06(1.36)	22.11(0.69)	14.63(0.57)
TS	68.93(0.7)	41.35(0.98)	20.58(1.86)	28.93(1.23)	15.23(0.28)	9.8(0.64)
RNN	**73.39(0.78)**	54.21(2.94)	30.51(0.8)	31.63(0.12)	21.19(0.87)	11.81(0.94)
TI	73.01(0.86)	58.81(1.67)	32.22(0.89)	31.9(0.82)	21.46(0.37)	12.78(0.77)
IMPORT($= 0$)	72.66(0.96)	57.62(0.75)	31.36(4.55)	30.76(0.7)	21.53(2.07)	11.41(0.46)
IMPORT(> 0)	73.13(0.62)	**61.38(0.62)**	**42.29(3.08)**	**32.44(1.12)**	**24.83(0.87)**	14.47(2.05)

Table 1. Results on the Bandit problems with $\rho = 0.05$ and $T = 100$ (after $4 \quad 10^7$ environment steps). On average, the distribution over the arms changes 5 times per episode.

Method	Test reward		
	$= 0$	$= 0.05$	$= 0.1$
TS	93.54(1.99)	85.33(6.28)	91.34(4.28)
RNN	81.3(12.7)	73.6(6.3)	76(4.1)
TI	90.4(0.5)	83.6(2.4)	80.1(2.2)
IMPORT	**96.4(2.9)**	**93.4(3.2)**	**97.2(1.5)**

Table 2. CartPole with task embeddings

task descriptor.

We report results obtained in a (non-stationary) multi-armed bandits (MAB) problem and on a classical control problem (CartPole). Examples of other environments are provided oin the appendix. Note that train/validation/test task sets are disjoint to support a). To test d), we study CartPole for two types of privileged information: 1) summarizes the dynamics of the system, e.g. pole length and 2) encodes a task index (one-hot vector) in a set of training tasks. When using 2), we restrict the size of the training set to be small, thus making generalization property even harder.

Multi-armed bandits: We use MAB problems with K arms where the reward of each arm k follows a Bernoulli distribution with parameter $_k$. In the non-stationary case, there is a probability to re-sample the value of $= (_k)_{k=1}^K$ at each time step. One random arm is associated with Bernoulli parameter $_k = _{max}$ and all other arms k have $_k \quad U([0; _{min}])$. Episode length is $T = 100$. In this environment, TS and TI are using a Beta distributions to model . We add bandit-specific baselines: *UCB* (Auer, 2003) and one of its non-stationary counter-part *Sliding-Window UCB* (Garivier & Moulines, 2008).

Table 1 shows that IMPORT learns complex probing behaviours and outperforms other methods on most settings. Moreover, the supervised auxiliary loss (> 0) really helps achieving better performance.

Control Environments: We use the *CartPole* environment from OpenAI Gym (Brockman et al., 2016), where controls physical variables of the system, e.g., the weight of the cart, the size of the pole, etc. The size of is 5 and the values of are normalized in $[1; 1]$. They are uniformly sampled at the beginning of each episode, and are resampled at each timestep with a probability . The maximum episode length is $T = 100$. We also evaluate sensitivity to task descriptors by considering the CartPole environment with task identifiers and R values of at train time, the validation being performed on episodes generated with 100 different values of , and the performance being reported over 100 other values. is encoded as a one-hot vector of size R reflecting which task the agent is currently playing.

Table 2 and Figure 2 shows IMPORT outperforms other models and better generalizes even with few tasks at train time. All results are presented in Appendix B.2 and B.3, as well as experiments on *Acrobot* that showcase same conclusions. Interestingly, IMPORT's decreasing performance after $4 \quad 10^6$ steps is overfitting on training tasks. TS fails as it predicts using a multinomial distribution; at test time, it behaves according to informed policies $^{\wedge}_t$ where $^{\wedge}_t$ corresponds to the index of one training task.

5. Conclusion

We have proposed a new policy architecture for learning in multi-task environments. IMPORT is trained only on the reward objective, and leverages the informed policy knowledge to discover a good trade-off between exploration and exploitation. It is thus able to learn better strategies than Thompson Sampling approaches, and faster than RNN and Task Inference policies. Moreover, our approach works well also in non-stationary settings and is able to adapt to generalize to tasks unseen at train time. In a near future, we will investigate the use of a scheduling to not waste too many samples once the informed policy is learned.

References

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null): 397–422, March 2003. ISSN 1532-4435.

Bakker, B. Reinforcement learning with long short-term memory. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 1475–1482, Cambridge, MA, USA, 2001. MIT Press.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv e-prints*, art. arXiv:1703.03400, Mar 2017.

Garivier, A. and Moulines, E. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. *arXiv e-prints*, art. arXiv:0805.3415, May 2008.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. A. Learning an embedding space for transferable robot skills. In *ICLR (Poster)*. OpenReview.net, 2018.

Heess, N., Hunt, J. J., Lillicrap, T. P., and Silver, D. Memory-based control with recurrent neural networks. *CoRR*, abs/1512.04455, 2015. URL http://arxiv.org/abs/1512.04455.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Whye Teh, Y., and Heess, N. Meta reinforcement learning as task inference. *arXiv e-prints*, art. arXiv:1905.06424, May 2019.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

Lazaric, A. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

Mnih, V., Puigdomènech Badia, A., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1602.01783, Feb 2016.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. Deep exploration via bootstrapped DQN. In *NIPS*, pp. 4026–4034, 2016.

Osband, I., Roy, B. V., Russo, D. J., and Wen, Z. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. *arXiv e-prints*, art. arXiv:1909.09157, Sep 2019.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019.

Taylor, M. E. and Stone, P. An introduction to inter-task transfer for reinforcement learning. *AI Magazine*, 32(1): 15–34, 2011. URL http://www.cs.utexas.edu/users/ai-lab?AAAIMag11-Taylor.

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017. URL http://arxiv.org/abs/1707.04175.

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

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 1015–1022, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273624. URL https://doi.org/10.1145/1273496.1273624.

Zhou, W., Pinto, L., and Gupta, A. Environment probing interaction policies. 2019.

A. Implementation details

A.1. Results preprocessing

We run each method for different hyperparameter configurations, specified in Appendix B, and choose the best hyperparameters using grid-search. We separate task sets into disjoints training, validation and testing sets. During training, every 10 model updates, the validation performance is measured by running on 100 episodes with taken from the validation tasks. Similarly, the test performance is measured using 100 testing tasks.

Each pair (method, set of hyperparameters) was trained with 3 seeds. For each method, we define the best set of hyperparameters as follows. First, for each seed, find the best validation performance achieved by the model over the course of training. The score of a set of hyperparameters is then the average of this performance over seeds. The best set of hyperparameters is the one with maximum score.

Plots. Each curve was obtained by averaging over 3 seeds already-smoothed test performance curves. The error bars correspond to standard deviations. Smoothing is done with a sliding window of size 11. For each method, we only plot the method with the best set of hyperparameters, as defined above. The x-axes of the plots correspond to environment steps.

Tables. Results in Tables 1, 2, 5 and 4 correspond to the mean and standard deviation (over seeds) of the test performance obtained by the policy extracted from the model with the best set of hyperparameters at maximum validation performance.

A.2. Hyperparameters

Hyperparameters ranges specified in Table 3 are kept constant on all environments. Environment-specific hyperparameters (hidden size hs, belief distribution for TS/TI, ...) will be specified in Appendix B.

Hyperparameter	Considered values
bs	4
	0.95
clip gradient	40
	$f1e^{-3}; 3e^{-3}g$
$_h$	$f1e^{-1}; 1e^{-2}; 1e^{-3}g$
$_c$	$f1; 1e^{-1}; 1e^{-2}g$
	$f1; 1e^{-1}; 1e^{-2}; 1e^{-3}g$
	$f0:5; 0:75g$

Table 3. Hyperparameters range

B. Experiments

In this section, we explain in deeper details the environments and the set of hyper-parameters we considered. We add learning curves of all experiments to supplement results from Table 1, 2, 5 and 4 in order to study sample efficiency. Note that for all experiments but bandits, is normalized to be in $[\ 1; 1]^D$ where D is the task descriptor dimension.

B.1. Bandits

At every step, the agent pulls one of K arms, and obtains a stochastic reward drawn from a Bernoulli distribution with success probability $_i$, where i is the arm id. The goal of the agent is to maximize the cumulative reward collected over 100 steps. At test time, the agent does not know $= (\ _1; :::; \ _K)$ and only observes the reward of the selected arm.

is sampled according to the following multivariate random variable with constants $_{max}$ and $_{min}$ fixed beforehand:

an optimal arm i is sampled at random in $U(\llbracket 1; K \rrbracket)$
and $_i = _{max}$
$8i \neq i\ ; \ _i \quad U([0; \ _{min}])$

At each time-step, there is a probability to sample a new value of .

We consider different configurations of this generic schema with $= 0:05; \ K \ 2 \ f5; 10; 20g; \ _{max} \ 2 \ f0:5; 0:9g; \ _{min} = 0:1$.

All methods use $hs = hs \ = 32$ and the belief distribution is either a Beta distribution or Gaussian. Other hyperparameters are presented in Table 3.

Since the setting with $K = 5$ is fairly easy to solve, RNN, TI and IMPORT perform on par (see Fig. 3). TS performs worse as it is sub-optimal in non-stationary environments. For $K = 10$ (Fig. 4), IMPORT largely outperforms other methods. When $_{min} = 0:5$, the gap between the optimal arm and the second best can be small. The optimal policy does not necessarily stick to the best arm and learning is slower. When $K = 20$ (Fig. 5), learning is harder and the UCB baseline is better.

B.2. CartPole.

We consider the classic CartPole control environment where the environment dynamics change within a set $M \ (j \ j = 5)$ described by the following physical variables: gravity, cart mass, pole mass, pole length, magnetic force. Their respective pre-normalized domains are $[4:8; 14:8]; [0:5; 1:5]; [0:01; 0:19]; [0:2; 0:8]; [\ 10; 10]$. Knowing some components of might not be required to behave optimally. The discrete action space is $f \ 1; 1g$.

's are re-sampled at each step with probability 2 f0; 0:05; 0:1g. Episode length is T = 100.

All methods use hs = hs = 16 and the belief distribution is Gaussian. Other hyperparameters are presented in Table 3.

Figure 6 shows IMPORT's performance and sample efficiency is greatly superior to other methods. IMPORT(> 0) performs on par or worse than TI, which proves that IMPORT main advantage is the auxiliary supervised loss. TI performs dramatically worse, showing reconstructing the entire is not optimal.

B.3. CartPole-task

To study how the different methods deal with cases where no meaningful physical parameters of the system is available, as well as studying their performance on tasks that were not seen during training, we conduct a new set of experiments in the CartPole environment described below. In this new set of experiments, represents the task identifier of the considered -MDP. Here is a one-hot encoding of the MDP, thus containing no relevant information on the world dynamics. To assess generalization on unseen tasks, we consider a training task set of R different tasks where the underlying dynamics parameters are sampled in the same way than for the usual CartPole environment.Validation and testing task sets are then additional disjoints set of 100 tasks (thus, there is no overlap between train, validation and test task sets).

's are re-sampled at each step with probability 2 f0; 0:1g. Episode length is T = 100.

Considered hyperparameters in CartPole-task are the same than the ones in CartPole except the belief distribution is multinomial.

In stationary environments, all methods are roughly equivalent in performance (Figures 7, 8). Indeed, in control problems, there is no need of a strong exploration policy since the underlying physics can be inferred from few transitions. When the environment is non-stationary, IMPORT is significantly better than the baselines. In the end, these experiments suggest that, in the stationary setting, all methods are able to generalize to unseen tasks on that environment. In the non-stationary setting however, IMPORT significantly outperforms the baselines.

B.4. Acrobot

Acrobot consists of two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height. Environment dynamics are determined by the length of the two links, their masses, their maximum velocity. Their respective pre-normalized domains are [0:5; 1:5]; [0:5; 1:5]; [0:5; 1:5]; [0:5; 1:5]; [3 ; 5]; [7 ; 11]. Unlike CartPole, the environment is stochastic because the simulator applies noise to the applied force. The action space is f 1; 0; 1g. We also add an extra dynamics parameter which controls whether the action order is inverted, i.e. f1; 0; 1g, thus j j = 7.

Method	Test reward	
	$\rho = 0$	$\rho = 0.1$
TS	89.8(2.2)	49.2(4.)
RNN	79.8(9.4)	63.2(6.)
TI	85.2(7.5)	71.2(8.4)
IMPORT	**90.1(1.5)**	**88.3(1.)**

Table 4. CartPole: Generalization with task identifiers and $R = 100$. μ is a one-hot vector of size R encoding the id of each training task (after 10^7 environment steps).

Method	Test reward	
	$\rho = 0.01$	$\rho = 0.05$
RNN	-406.4(30.1)	-459.(57.5)
TI	-278.4(45.)	-223.6(36.3)
TS	-242.9(4.9)	-190.4(51.9)
IMPORT	**-112.2(10.8)**	**-101.7(11.1)**

Table 5. Acrobot in non-stationary settings with $T = 500$ (after 6 10^6 environment steps).

's are re-sampled at each step with probability 2 f0; 0:01; 0:05g. Episode length is 500.

All methods use hs = hs = 64 and the belief distribution is Gaussian. Other hyperparameters are presented in Table 3.

IMPORT outperforms all baselines in every settings (Fig. 9).

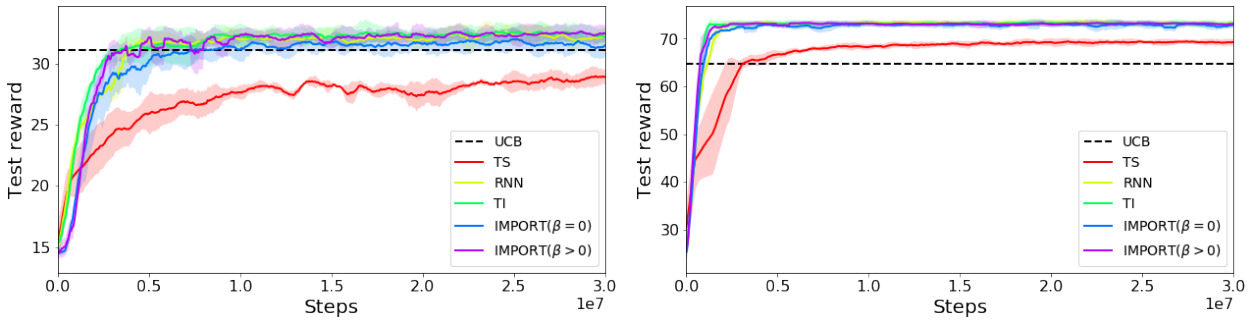

Figure 3. Training curves on the bandits environment with $K = 5$ and respectively $\mu_{min} = 0.5$ (left) and $\mu_{min} = 0.9$ (right).

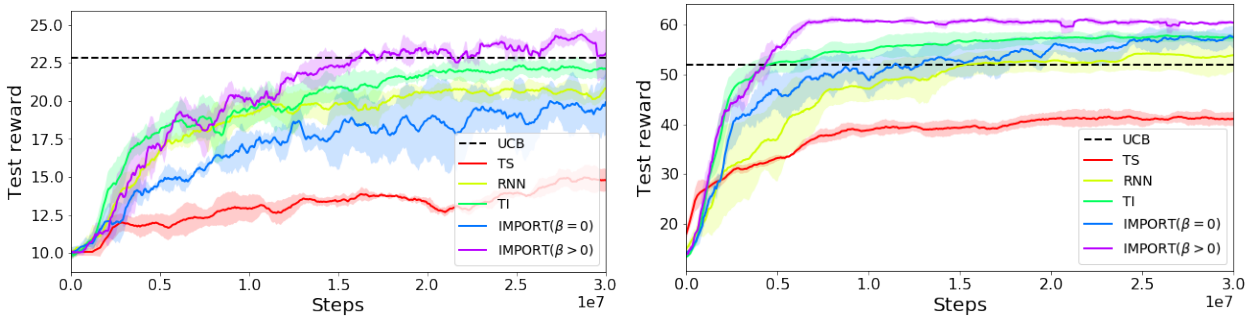

Figure 4. Training curves on the bandits environment with $K = 10$ and respectively $\mu_{min} = 0.5$ (left) and $\mu_{min} = 0.9$ (right).

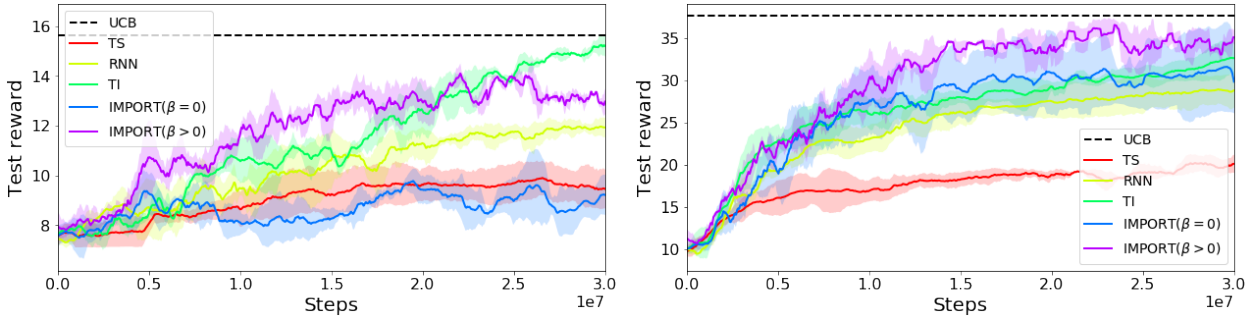

Figure 5. Training curves on the bandits environment with $K = 20$ and respectively $\mu_{min} = 0.5$ (left) and $\mu_{min} = 0.9$ (right).

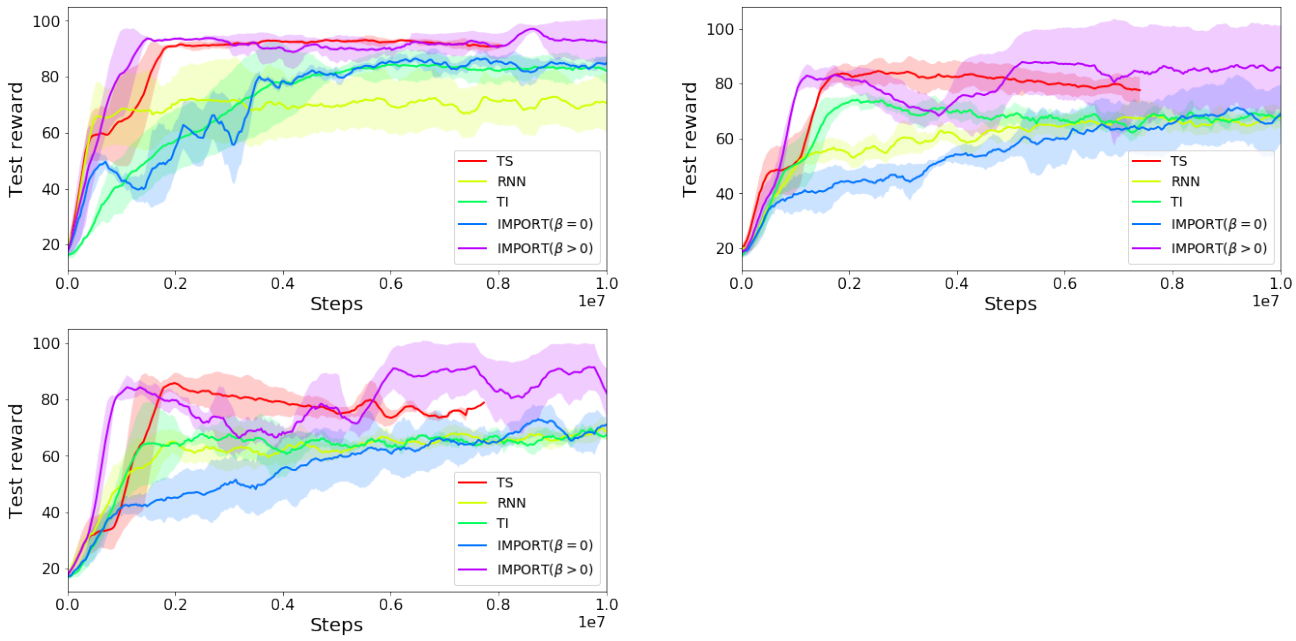

Figure 6. Training curves on the CartPole environment with respectively $\rho = 0$ (top left), $\rho = 0.05$ (top right), $\rho = 0.1$ (bottom)

.

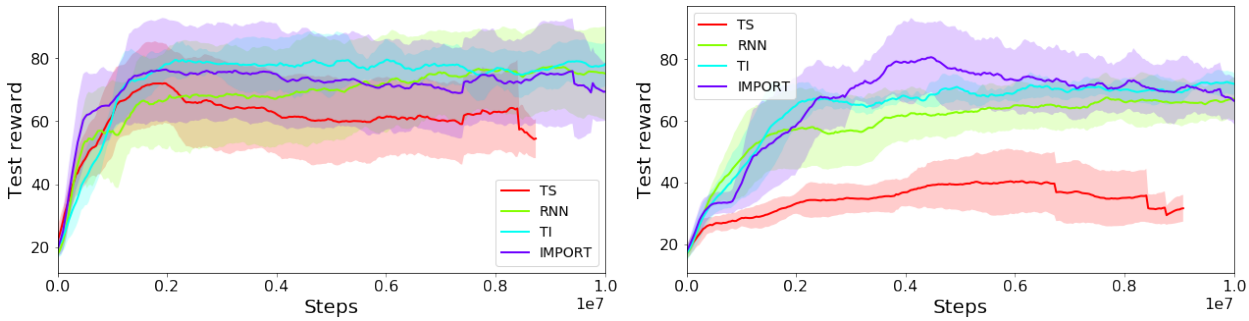

Figure 7. Training curves on the CartPole-task environment with $R = 10$ and respectively $\rho = 0$ (left), $\rho = 0.1$ (right)

.

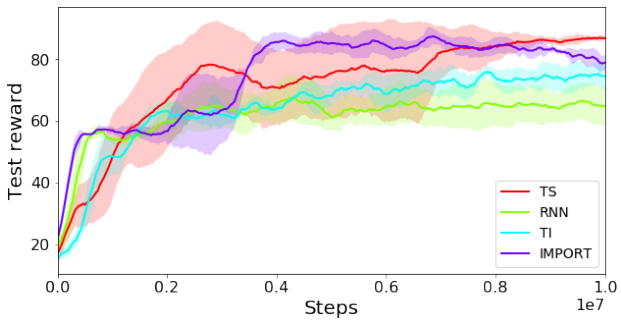

Figure 8. Training curves on the CartPole-task environment with $R = 100$ and respectively$\rho = 0$ (left), $\rho = 0.1$ (right)

.