POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMATICA E BIOINGEGNERIA
MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

[DRAFT] TRANSFER IN REINFORCEMENT LEARNING

Master Thesis of:
Andrea Sessa

Mat. **850082**

Supervisor:
Ing. Marcello Restelli

Co-supervisor:
Dott. Matteo Pirotta

Academic Year 2016 - 2017

Acknowledgements

THese are the acknowledgements
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Desidero inoltre ringraziare esplicitamente:

ESPLICITO1 per vari motivi;

ESPLICITO2 per altri motivi;

ESPLICITO3 per puro piacere, senza particolari motivi.

A. S.

Contents

List of Figures

Abstract

TExt of the abstract in english...

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: PoliMi, Master Thesis, LaTeX, Scribd

Chapter 1

Introduction

Reinforcement learning has demonstrated itself as one of the most promising branch of machine learning especially in the recent years.
RL success is mostly due to its applicability in large and complex control problem (robotics, economics, AI and so on) where the high complexity of the models involved makes impossible, for the human designer, to design a complete and efficient model. In all these situations RL techniques are able to provide a sufficiently accurate estimation of the model dynamics.
A generic RL algorithm permits an agent to *learn* a specific *task* by rewarding the agent when, during the training, it performs an action that bring it near to the goal or by punishing it when the action chosen bring it away from the goal (as described in Figure 1.1).

Many techniques exists to model such types of situations, one of the most popular and studied are the Markov Decision Processes (abbreviated as MDP). This formalism (we will give a more precise definition in chapter 2) permits to model the problem as a collection of states and actions. Moreover a reward function is also defined and used by the agent during the training phase.
When both the model of the problem, that is when the probability distribution given

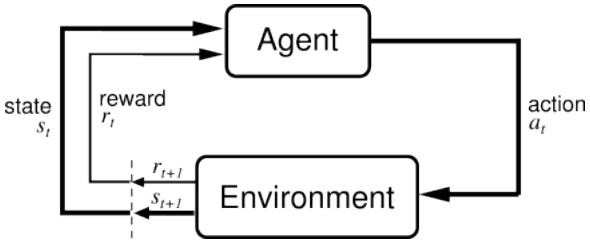

Figure 1.1: *A general learning agent*

a state-action pair is known to the designer of the problem, and when the reward function are known, a well established and efficient techniques are available to optimally solve the problem (eg. Dynamic programming).

Unfortunately in many real-world situations the model and the reward function associated to the MDP are not known and need to be learnt by the agent(*model-free*). In this context the term *"solving the problem"* acquires an additional meaning: we want still to determine the optimal policy but at the same time we want also learn the dynamics and reward function of the task. These two objective are not independent with each other instead they are strictly connected, indeed to learn the optimal policy it is necessary that the agent has learnt a good representation (in terms of dynamics and reward) of the problem. Others techniques are available in this context such as: Temporal difference, Q-Learning, SARSA, Fitted Q Iteration and so on.

A big problem in the model-free approach is the enormous amount of samples required to learn a good policy, this is true especially in many real world situation where the agent has a limited amount of time and resources to spend on the learning phase. Moreover every time the task at hand changes the entire learning process must be restarted from scratch even if very similar tasks have been already solved by the agent.

This problem motivates the need to have the possibility to transfer and reuse knowledge across different tasks. This idea naturally arise from psychology and cognitive science research(early 90s) where humans are shown to learn a specific task better and faster if they have already solved similar tasks in the past.
In the general machine learning framework the idea of the transfer of knowledge has already been successfully applied to problems in supervised learning(such as recommender systems, text classification, speech classification, etc.).
In the last 10 years the research on transfer learning has started to focus also on reinforcement learning applications.
In the RL field the main idea is that if the agent already knows the solution of set of *source tasks*, it can reuse that knowledge to speed up the learning of a new tasks(ie. the *target task*).
Every transfer RL algorithm must answer two questions: *what* and *when* to transfer?
The first question can be answered in different ways: one possibility to transfer instance of knowledge(ie. samples) from the source MDPs to the target MDP ([4]), another possibility is representation transfer where the transfer algorithm changes the representation of the solution of the source tasks ([7]), finally we have the possibility offered by parameters transfer where what we transfer is an initialization for the parameters of the learning algorithm of the target task (refer to [2]).
This work focuses on the transfer of instances.
The second question is a very important point: if we focus on the transfer of samples between tasks, the transfer should only occur when the source task presents some similarities with respect to the target task, if not a phenomena known in literature as *negative transfer* may occur. Such effect might actually worsen the performance of

the learning algorithm over the target task and therefore must be avoided as much as possible(a possible solution is presented in [4], the problem will be deeply discussed in this thesis).

When considering the problem of the transfer of knowledge across multiple tasks, a series of natural questions must be answered: Where to fetch the next sample? From which task? Should it be fetched from a source task or directly from the target task?. Methods have been proposed in the past, such has [4], where the authors introduce an algorithm which estimates the similarity between source and target tasks and selectively transfers from the source tasks which are more likely to provide samples similar to those generated by the target task. In this work we try to propose a novel approach based on a integration of the transfer learning framework with the *active learning* framework. Active learning tries to solve the same problem transfer learning solves. Many times the actual environment in which the agent lives is expensive, time-consuming and sometimes dangerous. The prohibitive cost of collecting samples in the real environment(ie. the target task) makes difficult (sometimes impossible) to learn a good policy. Active learning methods try to approach this problem by letting the algorithm itself to choose where, in the action-state space, to fetch the samples that will be used for training the agent.
Some approaches have been proposed in literature, such as [1], where the authors address the problem by focusing the sampling on the state-action pairs where the agent is more sensitive (ie. more uncertain).

Our approach differs from previous works: we try to integrate transfer learning and active learning ideas in a single algorithm. Precisely we seek a trade-off between the number of samples taken from the source tasks and the number of samples coming from the target tasks. Samples from the source tasks are cheap but they may negatively *bias* the agent in the target tasks(ie. negative transfer), on the other hand samples coming from the target task are expansive(they require the agent to perform trajectories over the real environment) but are unbiased since they come from the same problem the agent is trying to solve.

1.1 Mission and Contributions

The mission of this thesis is to produce a novel method to exploit the reuse of knowledge in the solution of new tasks. This is done by integrating transfer learning techniques with active learning techniques. At the same time we seek the optimal tradeoff between the number of samples coming from the source tasks and the number of samples coming from the target task. Moreover we aim to develop a new algorithm that exploits the active transfer of knowledge coming from old (already optimally solved) tasks and reuse it on new (not already solved) tasks. Finally we try to prove a series of theoretical bounds over the performance of the above mentioned algorithm. Moreover we also try to provide theoretical guarantees over the performance of the developed algorithm.

The contributions of this thesis can be summarized as follows:

- **Theoretical** contribution: We develop a new framework to permit the integration between active learning and transfer learning, this is done by seeking an optimal tradeoff between the amount of *old* and *new* knowledge.
- **Algorithmical** contribution: We introduce a new algorithm that applies in practice the active transfer of knowledge, moreover we prove a series of theoretical guarantees of the algorithm itself.

1.2 Contents Outline

This thesis is structured as follows:

- **Chapter 2** formally introduce the reader to the concept of Markov Decision Process and its application to reinforcement learning problems. In this chapter we introduce part of the notation that will be used in the rest of the thesis.
- **Chapter 3** provides an overview over the transfer learning framework including all the necessary mathematical notation. A short summary over the principal related works is included.
- **Chapter 4** proposes to the reader a brief introduction on the active learning framework. All the necessary mathematical notation is introduced, moreover a concise overview over related works is included.
- **Chapter 5** Algorithm
- **Chapter 6** Theoretical Results
- **Chapter 7** Experiments
- **Chapter 8** Conclusion/Discussion

Chapter 2

Reinforcement Learning

This chapter aims to introduce the reader to the problem of model-free Reinforcement Learning by providing a clear formalization of the main key concepts and the notations that will be used in the rest of the thesis.

2.1 Markov Decision Processes

In our context a task is identified as a discrete-time Markov Decision Process(abbreviated as MDP).
An MDP can be formalized as a tupe $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where:

- $\mathcal{S}$ is the set of the states(we assume it to be not finite).
- $\mathcal{A}$ is the set of all possible actions in every states(we assume it to be not finite)
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the transition model that assign to each state-action pair a probability ditribution(Π) over the state space.
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathbb{R})$ is the reward function that assign to each state-action pair a probability distribution over $\mathbb{R}$.
- $\gamma \in [0, 1]$ is the discount factor, which determine the sensibility of the agent toward future rewards.

At each time step the agent interact with the external environment according to its current policy, π: $\mathcal{S} \rightarrow \Pi(\mathcal{A})$, which given a state return a probability distribution over the action space.
How the agent can learn a policy? The objective of the agent is to maximize the expected sum of discounted rewards, that means to learn a policy $\pi*$ that leads to the maximization of the *utility* of the agent in each state of the MDP.
The concept of utility can be formalized introducing the definition of *value function*:

Definition 1. *The state–value function $V^{\pi}(s)$ of an MDP is the expected return starting from state s, and then following policy π*

$$V^{\pi}(s) = \mathbb{E}_{\pi}[v_t | s_t = s] \tag{2.1}$$

Suppose to consider a finite state-action space then the value function of the agent can be expressed according to the Bellman equation:

$$V^{\pi}(s) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V^{\pi}(s') \tag{2.2}$$

Given an MDP the agent is interested in learning a policy that maximizes $V(s)$ in all the states s, that is:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V^{*}(s') \right\} \tag{2.3}$$

where $R(s,a) = \mathbb{E}[\mathcal{R}(s,a)]$

However for control purposes, rather than the value of each state, it is easier to consider the value of each action in each state.

Definition 2. *The action–value function $Q^{\pi}(s,a)$ is the expected return starting from state s, taking action a, and then following policy π*

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[v_t | s_t = s, a_t = a] \tag{2.4}$$

Also the action-value function can be decomposed like in the case of the value function:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) Q(s', a') \tag{2.5}$$

Again the agent will try to learn a policy that maximizes Q^{π} overs the state-action space:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) = R(s,a) + \gamma \max_{a' \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) Q^{*}(s', a') \tag{2.6}$$

where $R(s,a) = \mathbb{E}[\mathcal{R}(s,a)]$

In this context a *deterministic* optimal policy can be obtained by choosing the action a that maximizes $Q(s,a)$ in each state s:

$$\pi(s) = arg \max_{a \in \mathcal{A}} Q^{*}(s,a) \tag{2.7}$$

Now we proceed to give similar definitions for the case in which both the action and state space are continuous (ie. contain an infinite number of states and actions).

Formally we assume state and action spaces to be complete, separable metric spaces $(\mathcal{S}, d_{\mathcal{S}})$ and $(\mathcal{A}, d_{\mathcal{A}})$ equipped with their $\sigma-$algebras, $\mathcal{B}(\mathcal{S})$ and $\mathcal{B}(\mathcal{A})$.
We consider infinite horizon problems where the future reward are discounted by γ.
The utility of each state following the policy π can be expressed with a modified version of the previous Bellman equation:

$$V^{\pi}(s) = \int_{\mathcal{A}} \left(\mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} V^{\pi}(s^{'}) \mathcal{P}(ds^{'}|s,a) \right) \pi(da,s) \tag{2.8}$$

Again, for convenience, we prefer to focus on the expected return of taking an action a in state s and following π:

$$Q^{\pi}(s,a) = \mathcal{R}(s,a) + \int_{\mathcal{S}} \int_{\mathcal{A}} \mathcal{Q}^{\pi}(s^{'},a^{'}) \pi(da^{'}|s^{'}) \mathcal{P}(ds^{'}|s,a) \tag{2.9}$$

The correspondent optimality equation (Q^{*} and V^{*}) can be obtained by maximizing respectively Q^{π} and V^{π}.
In the following we always assume to have continuous states and actions spaces.

2.1.1 Learning in continuous MDP

In this section we focus on the problem learning an optimal policy when both the states and actions spaces contain an infinite number of elements.
This situation makes infeasible the use of algorithms that use a tabular representation of the value(or action-value) function.
The problem is that there are too many states and/or actions to store in memory, moreover it is slow to learn the value of each state individually.

The simplest idea to solve a continuous MDP is to *discretize* both the action and state space in a manageable number of discrete states and actions and then solve the resulting MDP by using a standard policy/value iteration algorithm designed for finite MDPs. Despite the simplicity this type of approach may work well in many situations, however it has two main downsides:
First of all discretization represents a too *simple* approximation of the value function; indeed V^{π} is approximated with a piece-wise constant function; this approximation can not work well when the function to approximate is smooth.
The second disadvantage is related to the *curse of dimensionality*; suppose to discretize each dimension of $\mathbb{R}^n$ into k states, then the resulting number of states is k^n. So the number of discretized states grows exponentially with respect to the dimensionality of the original space.
These downsides makes often impossible the use of discretization in real-world problems.

More advanced techniques directly eliminate the need of a tabular representation of V^{π}. This approach, called *function approximation*, uses data coming from the MDP (or from a simulator) to produce a valid estimation of the value function.
For example, suppose we execute m trajectories over the MDP each of length T.

Each trajectory consists in a sequence of state/action pair:

$$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} \cdots \xrightarrow{a_{T-1}^{(1)}} s_T^{(1)}$$

$$s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} \cdots \xrightarrow{a_{T-1}^{(2)}} s_T^{(2)}$$

$$\cdots$$

$$s_0^{(m)} \xrightarrow{a_0^{(m)}} s_1^{(m)} \xrightarrow{a_1^{(m)}} \cdots \xrightarrow{a_{T-1}^{(m)}} s_T^{(m)}$$

We can now apply a generic learning algorithm to predict s_{t+1} as a function of the previous state s_t and the action a_t. In principle any function approximator can be used. However the choice may depend on some properties of the specific problem: experience may not be iid. (ie. timesteps may be correlated), rewards may be delayed (and not instantaneous), V^π may not be stationary during control, etc.
Known literature is divided in two main approach:

Incremental methods: Lets denote by V_ϑ the function approximator for V^π and let $L(\vartheta)$ a differentiable function of parameters ϑ defined as:

$$L(\vartheta) = \mathbb{E}_\pi[(V^\pi(s) - V_\vartheta(s))^2 | s_t = s] \tag{2.10}$$

The goal is to find a vector of parameters ϑ that minimizes $L(\vartheta)$, ie. the mean squared error between the true value function and the aproximator V_ϑ.
This, for example, can be done by following the opposite direction of the gradient of $L(\vartheta)$ ($\nabla L(\vartheta)$).

Batch methods: The main advantage of incremental methods is given by the simplicity behind the idea of gradient descent however many times these techniques are not sample efficient. On the other hand batch methods are not online methods; first the agent collects a sufficient amount of experience and then an optimization algorithm is applied to seek to the best fitting value function.
Precisely we define *experience* $\mathcal{D}$ as a collection of pairs $(state, value)$

$$\mathcal{D} = \{< s_1, v_1^\pi >, < s_2, v_2^\pi >, \cdots, < s_T, v_T^\pi >\} \tag{2.11}$$

Then the simplest idea is to find a vector of parameters ϑ that minimize the sum-squared error between $V^\pi(s_t)$ and the target values v_t^π

$$LS(\vartheta) = \sum_{t=1}^{T} (v_t^\pi - V_\pi(s_t))^2 \tag{2.12}$$

hence

$$\vartheta^* = arg \min_\vartheta LS(\vartheta) \tag{2.13}$$

2.1.2 Fitted Q-Iteration

Chapter 3

Transfer Learning

The key insight behind the idea of the transfer of knowledge comes from psychology: human beings learn a new task much faster if, in the past, they solved other similar tasks. Learning to drive a bike may facilitate learning to drive a car, learning to play organ may help in learning to play piano.
In the last years data mining and machine learning techniques have achieved significant success in many knowledge-engineering areas such as classification, regression and clustering.
However a large part of this techniques is based on the assumption that the training data and the test data come from the same distribution (ie. from the same task). The problem is that every time there is a significant difference between the two distribution the entire model must be learned again from scratch. In many real-world application it is impossible, every time the target task changes, to recollect all the needed data to train the new model. It would be nice to reutilize the knowledge (ie. data) collected for tasks similar to the target one.
A typical situation is when the data at hand may be easily outdated. In this case the data collected at the begin of a period may not follow the same distribution at the end of the same period. Transfer learning may help in avoiding the effort of rebuilding the model every time.
Another field of application of transfer learning that has been deeply investigated in the last years is Reinforcement Learning. In this situation the need of an efficient transfer of knowledge is even more evident. In many real-world scenario (such as Robotics or applications involving human interaction) the amount time needed to learn a good policy from samples coming from the target tasks if often prohibitive.

Transfer learning tries to reduce the learning time of an optimal policy by *reusing* knowledge coming from previously solved task (Figure 3.1).

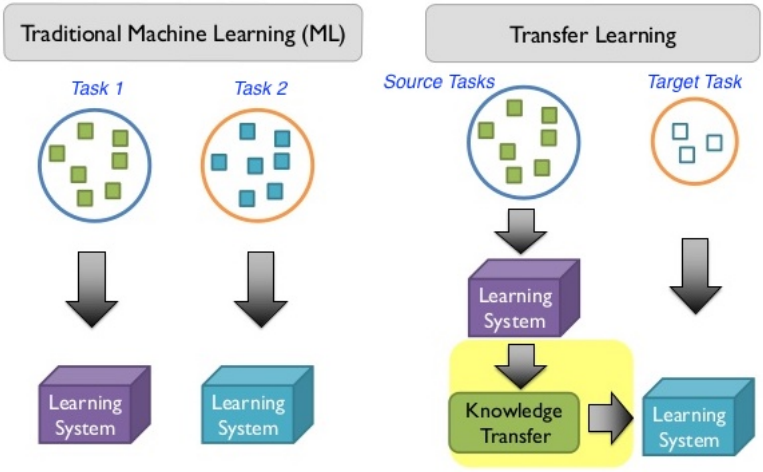

Figure 3.1: *Transfer Learning vs Traditional ML*

3.1 Related Works

Scientific research on transfer learning has attracted more and more attention since 1995.
Earlier works focuses on supervised learning mainly with application in classification and regression problems. For example [5] analyzes the problem of the transfer of knowledge under classification settings where the test and training distribution are mismatched. The authors propose a solution in which an auxiliary variable μ_i (for each sample $(\mathbf{x}_i^a, y_i^a)$ in the training dataset) is introduced to reflect the mismatch between the two distributions; when $(\mathbf{x}_i^a, y_i^a)$ is too different with respect the test distribution an appropriate choice of μ_i makes the final classifier less sensitive to the i$-th$ sample.
Another example, but applied to regression problems, can be found in [6]; here the authors try to develop one of the first general framework for transfer learning and analyze its correctness using the *Probably Approximately Correct (PAC)* learning theory. The paper proposes a modification of the classical boosting algorithm to successfully include samples coming from a mismatched distribution; the goal is to learn a high quality model using as much as possible samples coming from the auxiliary data source.
A more theoretical work is proposed by [8]; the authors explore a classification setting in which targets concepts are assumed to be drawn from a known distribution. The main goal was to study the total number of sample required to learn all targets to an arbitrary specified expected accuracy. The paper shows an interesting connection with the theory of active learning that will be discussed in the next chapter.

Only in the last 10 years research on transfer learning has focused also on reinforcement learning. Works that have been published differs in the type of knowledge transferred (instances, models representation or parameters) or in the number and domain of source and target tasks. In this thesis we mainly focus on the transfer of

instances from a set of source tasks to a single target task; considering this setting some interesting results are presented in [4] where the authors successfully show (empirically) that, when the samples coming from the source tasks are sufficiently *symilar* to ones already collected for the target task, the transfer of instances is possible and permits to significantly reduce the time needed to learn an optimal policy on the target task.
A more theory-oriented work appears in [3], this paper represents one of the first attempt to provide a theoretical formalization of the sample-transfer problem, in particular the authors provide a series of theoretical bounds for different transfer algorithm showing the potential of knowledge transfer in a single-task learning setting.

3.2 Notation and formal specification

In this section we provide a formal specification for the transfer learning framework used in this work.

Definition 3. *A task T is an MDP defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}_T, \mathcal{R}_T, \gamma)$.*

Definition 4. *An environment $\mathcal{E}$ is a pair $(\mathcal{T}, \Omega)$ where $\mathcal{T}$ is the task space and Ω is a probability distribution over the tasks in $\mathcal{T}$*

In our situation we have a set of source MDPs plus a target MDP: $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ where $\mathcal{M}_1, \ldots, \mathcal{M}_{n-1}$ are the source MDPs and $\mathcal{M}_n$ is the target MDP.
In our framework we focus on the transfer of samples between the source MDPs and the target MDP; a sample in our context is a tuple $(s, a, s^{'}, r)$ where s and $s^{'}$ are states, a is an action and r is the reward in $s^{'}$.
Our goal is to identify, at each time-step, which is the optimal sample to transfer from a source MDP to the target MDP. The transferred sample is optimal in the sense that, among the available samples, it is the one that carries the higher amount of informativeness. In particular we may be interested in transfer samples from the set of source tasks but also from the target task itself, that is we assume that if N_s is the number of samples available from the source MDPs than there are N_t samples available directly from the target task with $N_t << N_s$.
The algorithm identifying the best sample must also consider the origin(source or target) of the sample. Samples coming from the target task only help in reducing the uncertainty over the target task and they do not introduce further bias

Chapter 4

Active Learning

4.1 Related Works

4.2 Notation and Formal Specification

Bibliography

[1] Arkady Epshteyn, Adam Vogel, and Gerald DeJong. “Active reinforcement learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 296–303 (cit. on p. 11).

[2] Alessandro Lazaric. “Transfer in reinforcement learning: a framework and a survey”. In: *Reinforcement Learning*. Springer, 2012, pp. 143–173 (cit. on p. 10).

[3] Alessandro Lazaric and Marcello Restelli. “Transfer from multiple MDPs”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 1746–1754 (cit. on p. 19).

[4] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. “Transfer of samples in batch reinforcement learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 544–551 (cit. on pp. 10, 11, 19).

[5] Xuejun Liao, Ya Xue, and Lawrence Carin. “Logistic regression with an auxiliary data source”. In: *Proceedings of the 22nd international conference on Machine learning*. ACM. 2005, pp. 505–512 (cit. on p. 18).

[6] David Pardoe and Peter Stone. “Boosting for regression transfer”. In: *Proceedings of the 27th international conference on Machine learning (ICML-10)*. 2010, pp. 863–870 (cit. on p. 18).

[7] Satinder P Singh, Andrew G Barto, and Nuttapong Chentanez. “Intrinsically Motivated Reinforcement Learning.” In: *NIPS*. Vol. 17. 2. 2004, pp. 1281–1288 (cit. on p. 10).

[8] Liu Yang, Steve Hanneke, and Jaime Carbonell. “A theory of transfer learning with applications to active learning”. In: *Machine learning* 90.2 (2013), pp. 161–189 (cit. on p. 18).