POLITECNICO DI MILANO
Master Degree in Computer Science and Engineering Department
of Electronics, Information and Bioengineering

Cutting Back on MDPs' Features
A Theoretically Grounded Approach to Feature Selection in Reinforcement Learning

AIR Lab
Artificial Intelligence and Robotics Laboratory

Author:
Guido Dino Ballabio
ID 899545

Supervisor: Prof. Marcello Restelli
Co-supervisors: Dott. A. M. Metelli
Dott. M. Papini,
Dott. A. Tirinzoni

Academic Year 2018–2019

Ai miei genitori, a mia sorella,
ai miei amici e alla prof. Lentini

Abstract

The Reinforcement Learning (RL) framework has been under the spotlight in recent years, as it allows to design of controllers for previously intractable problems. Traditionally, RL has been applicable only to problems with low-dimensional state space. Although the use of Deep Neural Networks as function approximators with RL (DRL) has shown impressive results for the control of high-dimensional systems, DRL methods require a large amount of training samples to learn good policies even on simple environments, making them poor choice in real-world situations where sample collection is expensive.

The purpose of this thesis is the definition of a feature selection technique that enables to train RL agents on fewer dimensions. Moreover, the procedure should be able to discard not only useless features, but also low relevance features by using a theoretical bound to the error of the feature selection. The results are almost ideal in environments with limited dimensionality, but the bound relies on the Conditional Mutual Information of features, and the estimation of such information-theoretic quantity is still unreliable in high dimensional problems. Nonetheless, we provide novel theoretical insights into the control error in RL.

Contents

List of Algorithms

List of Tables

List of Figures

Chapter 1

Introduction

In this digital era, where pervasive information technologies proliferate in every aspect of life, large volumes of data are available for analysis. Large systems could use this data for automatic knowledge extraction in order to improve their performance through the application of one of the many techniques in the fields of supervised and unsupervised learning.

The above-mentioned fields of machine learning, while useful for prediction and analysis, do not provide a proper framework for automatic decision making. This functionality is necessary to reduce the reaction times of processes and improve the reliability, as well as the correctness, of the decisions.

A sub-field of machine learning was developed with the specific objective of addressing this family of problems where new challenges arise. This area goes under the name of Reinforcement Learning (RL): a computational approach that can perform automatic goal-directed decision-making.

RL can deal with large-scale or complex systems whose dynamics are non–linearly and even stochastically dependent on several state and control variables. In some of the hottest applications of RL, such as robotics, variables can be continuous data obtained from sensors. In other systems, typically those not made ad hoc by experts with hand-picked features, the actual relevance of the variables for the model is unknown.

The theoretical framework used to formalize the sequential decision making process is known as Markov Decision Process (MDP). Unfortunately, when the search space (state and control space) is high-dimensional, finding optimal solutions to MDPs is computationally intractable. As existing RL approaches perform poorly in such circumstances their range of applicability is reduced [23].

A long-studied idea to reduce the dimensionality of the MDPs is the state aggregation [32], whose aim is the reduction of the state space through the abstraction of clusters of similar states. Through this restriction of focus, during training, an agent's performance is expected to increase both in accuracy and training time.

The kind of aggregation that we consider here is executed by feature selection, where a reduced MDP model is built by taking into account only a subset of the original state variables. Thereby “states that differ only for the values of the removed variables are aggregated” [22]. As it is typical for large control systems to have a high dimensional action space as well, such as elevator groups control [19], we even consider the removal and aggregation of actions.

“Feature selection is thus becoming an increasingly prominent tool in the efforts of the RL community to expand the reach and applicability of RL” [21].

1.1 Goal and Motivation

The purpose of this thesis is the definition of a model-free offline filter feature selection technique that enables to train agents on fewer dimensions, thus reducing its overall complexity. Moreover, the technique should be able to discard not only useless features, but also low relevance features through an accurate estimate of an upper bound of the error introduced in the action-value function.

The source of inspiration for this thesis comes from a similar work [2] on feature selection in supervised learning, a different subfield of machine learning. This work applies its original idea, which constist of using information theoretical measures to develop an upper bound to the feature selection error, and extend it to the RL framework.

1.2 Proposed Solution

We introduce an upper bound to the feature selection error of the action-value function; this bound allows us to measure quantitatively the impact of the removal of features on an agent’s policy. The main idea is that the features orthogonal to the reward or conditionally independent of it given the selected features, at each time step, are irrelevant to the policy.

Whereupon it is possible to define a greedy algorithm for backward and forward feature selection which estimates the value of the bound and decides to stop with various strategies, of which the most notable is to stop just before a predefined error threshold is exceeded.

As a side contribution we adapted and implemented a correction of a well-known mutual information estimator for the conditional case.

1.3 Thesis Structure

The remainder of the thesis is structured as follows:

- Chapter 2

provides the necessary background information: the RL theoretical framework and formal definitions of information-theoretical measures on which the work is based.

- Chapter 3

  summarizes the state of the art in the field of feature selection in the RL framework and the fundamental papers from which this work derives.

- Chapter 4

  presents the theoretical bound, its derivation and the motivation behind its final form.

- Chapter 5

  introduces the proposed algorithms for the filter feature selection, possible optimization, and their computational complexity.

- Chapter 6

  presents the experiments with the relevant technical details and decisions, shows the results and finally explains them by presenting the limitations of this approach.

- Chapter 7

  summarizes and concludes the work done in the thesis, and presents final observations as well as directions for future work.

Chapter 2

Background

In this chapter we introduce the basic concepts about Markov Decision Processes (MDP), as formalization of the Reinforcement Learning (RL) problem. Later on we present some definitions of Information Theory (IT). The first and second sections are mostly based on the new edition of the classic [10], the third one on [33].

2.1 Reinforcement Learning

As [10, p. 1] stated "Reinforcement learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal.".

The terms supervised learning and unsupervised learning would seem to exhaustively classify machine learning paradigms, but they do not. Uncovering structure in an agent's experience can certainly be useful in reinforcement learning, but by itself does not address the reinforcement learning problem of maximizing a reward signal. We therefore consider reinforcement learning to be a third machine learning paradigm, alongside supervised learning and unsupervised learning and perhaps other paradigms.

Reinforcement Learning is a branch of machine learning that deals with sequential decision making by an agent in a given environment. In sequential decision making problems the goal of the agent is to select actions to maximize cumulative rewards where actions may have long–term consequences, hence reward may be delayed.

The agent, in order to improve, must learn by interacting with the environment. This is a major difference w.r.t. the rest of the machine learning subfields: in this framework the data are sequences of interactions.

One of the challenges that arise through interaction, and not in other kinds of learning, is the trade-off between exploration and exploitation. To obtain a high reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before.

Of all the forms of machine learning, reinforcement learning is the closest to the kind of learning that humans and other animals do.

It must be noted that RL hinges on the Reward Hypothesis as stated by Sutton [10].

Assumption 2.1.1 (Reward Hypothesis)**.** *That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).*

This is the reason why the reward in RL is scalar (a least in its classic definition). An assumption that, even though simplistic, seems to have stood the test of time.

Henceforth the formal definition of an RL problem must include three aspects: observation, action and a scalar reward. Also RL has its roots in optimization, dynamical systems and control theory, so the problem can be naturally formalized in terms of optimal control of a special class of discrete-time problems that model this kind of agent-environment interaction: MDPs.

2.2 Markov Decision Process

MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and, through those, future rewards.

In MDPs an environment is modeled as a set of states and actions that can be performed to control the system's state: at each time step t an agent executes an action a_t on the environment and perceives an observation o_t as well as a reward r_t.

As regards the state of the environment in MDPs, it is the same observed by the agent, which has an own internal representation of state equal to the environment's one. However RL can be applied also to the most general case of hidden information for which a different formalism is considered: Partially Observable MDPs (POMDPs).

Specifically to MDPs the environment must hold the Markovian property:

Definition 2.2.1 (Markov Property)**.** The next state of the environment does not depend on the history of interaction, but only on the current state and action.

It can be defined in a mathematical way as:

$$P\left(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0\right) = P\left(s_{t+1}|s_t, a_t\right) \tag{2.2.1}$$

We can then proceed to formally define a MDP.

Definition 2.2.2 (Markov Decision Process)**.**
A MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma, \mu)$ such that:

1. $\mathcal{S}$ is the set of states of an environment.
2. $\mathcal{A}$ is the set of actions an agent can take in the environment.
3. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a reward function.
4. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition probability function, otherwise represented as $P(s_{t+1} = s'|s_t = s, a_t = a)$ given two states $s, s' \in \mathcal{S}$ and an action $a \in \mathcal{A}$, subject to the Markov property.
5. $\gamma \in [0, 1]$ is the discount rate that weights the reward at each time step in the sum of rewards to be maximized.
6. $\mu : \mathcal{S} \rightarrow [0, 1]$ is the probability distribution of the initial state.

A MDP, and thereafter an RL problem, is solved when we have an optimal mapping between states and action to be taken in that state. This mappings, in general stochastic, are called policies, denoted by π.

Definition 2.2.3 (Policy)**.** A policy is a mapping from state to actions in probability, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$.

The last piece of information necessary to formally define the problem is the cumulative reward. For many reasons, including mathematical easiness one, it usually includes a discount factor that weights more immediate rewards than future ones. Hence the discounted sum of rewards since time step t, as a random variable, is:

$$J_t^\pi = \sum_{k=t}^{T} \gamma^k R_k^\pi \tag{2.2.2}$$

where R_k^π is the reward obtained at the k time step under policy π and T is a final time step,. When the task is a continuing one $T \rightarrow \infty$.

The optimal policy, denoted as π^* the final goal of RL, is the one maximizing the *expected cumulative reward*:

$$G_t = \mathop{\mathbb{E}}_{s\sim\mu} \left[\mathbb{E}\left[J_t^\pi | s_t = s\right]\right] = \mathop{\mathbb{E}}_{\mu} \left[J_t^\pi\right] \tag{2.2.3}$$

Here we define some technical terms that we will use later on: an *episode* is a realization from initial to final state of a MDP and its sequence of $(s, a, r) \in \mathcal{S} \times \mathcal{A} \times \mathcal{R}$ is called a *trajectory.*

Bellman equations

Two formulas have been defined to describe and theoretical solve MDPs in the field of dynamic programming: the Bellman equations, designed to exploit the optimal substructure of MDPs. They recursively split the computation of the expected total reward in the sum of the immediate reward and the value (expected total reward) from the next state.

In order to derive them we need to define two value function: the value of a state is defined by the state-value function.

Definition 2.2.4 (State-Value function)**.** Given a state s at time step t, the value function $V^\pi : \mathcal{S} \to \mathbb{R}$ returns the expected total reward from that state onward following policy π.

$$V^\pi(s) = \mathbb{E}\left[J_t^\pi | s_t = s\right] \tag{2.2.4}$$

Similarly we can define the value of taking an action in a state:

Definition 2.2.5 (Action-Value function)**.** Given a state s and an action a at time step t, the value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ returns the expected total reward from taking that action in that state onward following policy π.

$$Q^\pi(s,a) = \mathbb{E}\left[J_t^\pi | s_t = s, a_t = a\right] \tag{2.2.5}$$

Exploiting the recurrence relation $J_t^\pi = R_t + \gamma J_{t+1}^\pi$ in the above definition we can easily obtain the Bellman equations:

$$\begin{aligned}
V^\pi(s) =& \mathbb{E}\left[J_t^\pi | s_t = s\right] \\
=& \mathbb{E}\left[R_t + \gamma J_{t+1}^\pi | s_t = s\right] \\
=& \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \Big[\mathcal{R}(s,a,s') + \\
& \qquad\qquad + \gamma \mathop{\mathbb{E}}_{\pi}\left[G_{t+1} | s_{t+1} = s'\right] \Big] \\
=& \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \Big[\mathcal{R}(s,a,s') + \gamma V^\pi(s') \Big]
\end{aligned} \tag{2.2.6}$$

which can also be written with a compact matrix notation:

$$V^\pi(s) = R + \gamma P^\pi V^\pi \tag{2.2.7}$$

An analogous equation exists for the action-value function:

$$\begin{aligned}
Q^\pi(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \Big[& \mathcal{R}(s,a,s') + \\
& + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^\pi(s',a') \Big]
\end{aligned} \tag{2.2.8}$$

It must be noted that V^π can always be retrieved from Q^π by:

$$V^\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi(a|s)}\left[Q^\pi(s,a)\right] \quad \forall s \in \mathcal{S} \tag{2.2.9}$$

The goal of RL of finding policy π^* can then be stated using the state-value function:

$$V^{\pi^*}(s) = \sup_{\pi} V^{\pi}(s) \quad \forall s \in \mathcal{S} \tag{2.2.10}$$

however, it is hard to retrieve a policy from a V-function so it is usually rewritten with the Q-function, which is better suited for control problems:

$$Q^{\pi^*}(s,a) = \sup_{\pi} Q^{\pi}(s,a) \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \tag{2.2.11}$$

and then compute the optimal deterministic policy (that always exist) from it:

$$\pi^*(s) \in \operatorname*{argmax}_{a} Q^{\pi^*}(s,a) \quad \forall s \in \mathcal{S} \tag{2.2.12}$$

with an abuse of notation for deterministic policies such that $\pi : \mathcal{S} \to \mathcal{A}$ instead of the usual stochastic formulation.

Finally, the stationary visitation distribution over the states given an initial distribution μ and a policy π is $\frac{d^{\pi}_{\mu}}{1-\gamma}$ and can be seen as the solution to the evaluation of a Bellman expectation equation eq. (2.2.7):

$$d^{\pi}_{\mu} = (I - \gamma P \pi)^{-1} \tag{2.2.13}$$

where $V^{\pi} = d^{\pi}_{\mu} R$.

2.3 Overview of Solutions

We will briefly categorize the known solutions methods used in RL.

Firstly, there is a distinction between algorithms that use (and build) a model of the environment and those which do not.

Methods for solving RL problems that use models and planning are called *model-based* methods, as opposed to simpler *model-free* methods that are explicitly trial-and-error learners, viewed as almost the opposite of planning.

Model-free methods can be applied straightforwardly to any reinforcement learning problem since they do not require any model of the environment, hence they provide the more general solution.

Model-based methods try to utilize some information of the environment for planning. Planning with a model can substantially improve the sample efficiency of the algorithm. However to generalize this method the focus has to be on learning a model of the environment by performing experiments on the system.

In this thesis we focus on model-free RL, so we subdivide them even more.

Most model-free approaches either try to learn a value function and infer an optimal policy from it (Value function based methods) or directly search in the space of the policy parameters to find an optimal policy (Policy search methods).

Model-free approaches can also be classified as being either on-policy or off-policy [18]. On-policy methods use the current policy to generate actions and use it to update the current policy while off-policy methods use a different exploratory policy to generate actions w.r.t. the policy that is being updated.

2.4 Information Theory

When Shannon defined entropy it was part of his attempt to establish “A Mathematical Theory of Communication” [39]. It turned out that Shannon’s work was the key concept for a theory that has far more possible fields of application than just communication theory: information theory (IT). “Elements of Information Theory” [33] gives an extensive overview of the application of Shannon’s information theory that ranges from the original field of communication theory to computer science and from which we draw this chapter.

We present the fundamental quantities of IT. For simplicity, we provide the definitions for continuous scalar random variables, although all these concepts straightforwardly generalize to discrete variables (but not all of their properties when noted) and to vectors of random variables.

Entropy

Definition 2.4.1 (Entropy)**.** The entropy $\mathrm{H}(X)$ of a random variable X, having p as probability density function, is a common measure of uncertainty:

$$\mathrm{H}(X) = -\mathbb{E}_X[\log p(x)] = -\int_x p(x)\log p(x)\mathrm{d}x. \tag{2.4.1}$$

Since entropy was first introduced in communication theory, it is usually expressed in bits and the base of the logarithm is 2. A notable property of the entropy of a *discrete* random variable is non-negativity as it is density is bounded: $p(\mathbf{x}) \leq 1$. In this case entropy can be seen as an absolute value of the information content of the random variable, however, these considerations do not hold for continuous r.v..

It must be noted that if the random variable is not scalar, as it is in the general case, then it is also referred to as *joint entropy*, which could be defined for the joint distribution of any r.v..

As a matter of fact entropy is defined for any probability distribution, also conditional entropy.

Definition 2.4.2 (Conditional Entropy)**.** The conditional entropy $\mathrm{H}(X|Y)$ of a random variable X given Y measure the uncertainty (or information) of X already knowing Y:

$$\begin{aligned}\mathrm{H}(X|Y) = -\mathbb{E}_{X,Y}[\log p(x|y)] = -\mathbb{E}_{X,Y}[\log p(x,y)] +\\ -\mathbb{E}_{X,Y}[\log p(y)] = \mathrm{H}(X,Y) - \mathrm{H}(Y)\end{aligned} \tag{2.4.2}$$

Probability metrics

This quantities however do not enable us to compare r.v. (or distributions) for that we need some distance functions so called probability metrics. Here we introduce just two of the many available [35].

Definition 2.4.3 (Kullback-Liebler Divergence)**.** The Kullback-Liebler divergence (KL) of two probability density functions p and q on the same support, $D_{\mathrm{KL}}(p||q)$, describes the amount of information gained by knowing the true distribution of a random variable p instead of assuming its distribution to be q.

$$D_{\mathrm{KL}}(p||q) = \mathbb{E}_X\left[\log\frac{p(x)}{q(x)}\right] = \int_X p(x)\log\frac{p(x)}{q(x)}\mathrm{d}x \tag{2.4.3}$$

The KL divergence is also known as Relative Entropy for obvious reasons. Moreover, as entropy is not an absolute measurement of information for *continuous* r.v., the KL is typically used as main measure of information for them. In fact it was firstly presented as an extension of Shannon's Entropy.

It must be noted that Relative Entropy is not a proper metric: it is not symmetric nor satisfies the triangle inequality, however it is always non-negative for both *continuous* and *discrete* r.v.: $D_{\mathrm{KL}}(p||q) \in [0, \infty]$.

Definition 2.4.4 (Total Variation Distance)**.** The total variation distance (TV) of two probability density functions p and q on the same support, $D_{\mathrm{TV}}(p||q)$ is, informally, the largest possible difference between the probabilities that the two probability distributions can assign to the same event.

$$D_{\mathrm{TV}}(p||q) = \sup_{A\subseteq X}|p(A) - q(A)| = \frac{1}{2}\int_X |p(x) - q(x)|\mathrm{d}x \tag{2.4.4}$$

From the definition itself it is clear that $D_{TV} \in [0, 1]$.

The TV distance is an exceptionally strong notion [30] hence usually it is related to the KL through a bound:

Definition 2.4.5 (Pinsker's inequality)**.** A bound tight up to a constant factor of TV and KL:

$$D_{\mathrm{TV}}(p||q) \leq \sqrt{\frac{1}{2}D_{\mathrm{KL}}(p||q)} \tag{2.4.5}$$

Mutual Information

The KL divergence in particular can be used to measure the information a random variable contains about another random variable, i.e., their mutual information.

Definition 2.4.6 (Mutual Information)**.** The mutual information (MI) $I(X;Y)$ of two r.v. X and Y is a measure of the uncertainty explained by X about Y and vice-versa:

$$\begin{aligned} \mathrm{I}\left(X;Y\right) = D_{\mathrm{KL}}\left(p(x,y)||p(x)p(y)\right) = \\ = \iint_{X,Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \mathrm{d}x\mathrm{d}y \end{aligned} \tag{2.4.6}$$

Furthermore MI can be related to entropy:

$$\mathrm{I}\left(X;Y\right) = \mathrm{H}\left(X\right) - \mathrm{H}\left(X|Y\right) = \mathrm{H}\left(X\right) + \mathrm{H}\left(Y\right) - \mathrm{H}\left(X,Y\right) \tag{2.4.7}$$

in this formulation it can be easily seen that MI is both symmetric and non-negative, recalling that $\mathrm{H}\left(X,Y\right) \leq \mathrm{H}\left(X\right) + \mathrm{H}\left(Y\right)$ and that equality holds only when the r.v. are independent. Another formulation of MI will be useful later on in relation to conditional probabilities:

$$\mathrm{I}\left(X;Y\right) = \mathbb{E}_X\left[D_{KL}\left(p(y|x)||p(y)\right)\right] \tag{2.4.8}$$

This definition can be straightforwardly extended by conditioning on a third random variable, obtaining the conditional mutual information [2].

Definition 2.4.7 (Conditional Mutual Information)**.** The conditional mutual information (CMI) $I(X;Y|Z)$ of two r.v. X and Y given Z, is a measure of the uncertainty explained by X about Y and vice-versa, given that Z is known:

$$\mathrm{I}\left(X;Y|Z\right) = \mathbb{E}_Z\left[\mathrm{I}\left(X|Z;Y|Z\right)\right] \tag{2.4.9}$$

$$\text{eq. (2.4.8)} = \mathbb{E}_Z\left[\mathbb{E}_X\left[D_{\mathrm{KL}}\left(p(y|x,z)||p(y|z)\right)\right]\right] \tag{2.4.10}$$

$$= \iiint_{X,Y,Z} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)} \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{2.4.11}$$

Conditioning can increase or decrease information but CMI maintains the same properties of MI. Moreover MI has an easier formulation on which to work, known as chain rule:

$$\mathrm{I}\left(X;Y|Z\right) = \mathrm{I}\left(X;Y,Z\right) - \mathrm{I}\left(X;Z\right) \tag{2.4.12}$$

This quantity is perfectly suited for giving a score of relevancy and redundancy to variables in relation to a specific one.

Chapter 3

State of the Art

In this chapter we discuss the state of the art of feature selection (FS) procedures for MDPs in the field of reinforcement learning.

3.1 Feature Selection in Machine Learning

All machine learning fields incur in the curse of dimensionality: the exponential increase in volume of the input space as the number of dimensions (d) gets larger. The effects of the curse are multiple: the major ones are the exponential increase ($\propto d$) of the computational cost that at the very least increases because more data needs to be stored and read for each sample and of the number of data points to cover the sample space, which can lead to a large variance and thus overfitting.

FS is a way to deal with the curse of dimensionality by choosing a subset of all the features whose cardinality is the new (shrunk) dimensionality of the problem. In practice, these approaches do require that good variables are present within the larger set [14], possibly under the assumption that the feature that embed the MDP, the one that characterize its transition and reward functions, are contained in the complete feature set available.

FS in *supervised learning* has a long history. In RL, on the other hand, feature selection is a less mature field, but has received a lot of attention recently, both in theoretical and practical works.

Various FS strategies have been explored so far in supervised learning, which can be categorized into the filter, wrapper and embedded approaches [11]:

1. Filter methods rank features based on their intrinsic properties, such as their "relevance" and/or "redundancy" measured by a proper statistic and thus are independent of the model.

2. In the wrapper heuristics, features are selected depending on the subsequent learning process, i.e., based on the learner performance. As

such a try to solve the actual problem of maximizing accuracy in the final task. However they are more computationally expensive due to the cross-validation process.

3. Embedded approaches are alike to wrapper methods since they share the objective of maximizing the performance of a learning algorithm or model. The difference to wrapper methods is that an intrinsic model-building metric is used during learning and exploited for variable selection.

As many FS methods for RL were inspired by the supervised learning literature, they roughly follow the same subdivision and we present them by category.

3.2 Embedded Methods

First we address the embedded approaches for historical purposes, as these are not of any use with different algorithms and thus they are just examples of how to reduce the state space of MDPs.

The most notable embedded methods draw inspiration from a renowned supervised method: LASSO. LASSO applies L_1 regularization to linear regression to find a sparse solution. In an analogous way [26] finds sparse solution in the RL framework: they apply L_1 regularization to a least-square fitting to the value function of a linear approximator: least-square temporal difference (LSTD). This algorithm, least-angle regression for TD (LARS-TD), needed a complex custom solver so it was later improved by [24] in order to use a simpler one, by reducing the problem to a linear complementarity one (LC-TD).

For algorithms based on similar regularization techniques [25], upper bounds to the error with respect to the optimal value function were presented. However, all of this approaches are sound only for linear value-function approximation and thus are not generalizable.

3.3 Wrapper Methods

Automatic FS in RL "has typically focused on using a linear value function approximation method with a feature selection wrapper" [14].

Indeed, as mentioned above, most of the results in FS are are based on linear value approximation and how to optimize feature selection for it. Consequently, also for wrapper methods, all theoretical guarantees developed in literature hold only in the linear case. Moreover, while these wrapper methods methods would be useful for specific RL frameworks such as policy iteration, they may not be directly employed in other frameworks such as policy search [23].

The most venerable example of these are the decomposition of bellman error in reward and model error, as in [29]. It must be noted, though, that their use of this technique is actually more akin to feature construction, which builds feature combinations given a subset of raw ones instead of selecting them.

Orthogonal Matching Pursuit

Many works about FS were inspired by such decomposition, notably an adaptation of orthogonal matching pursuit (OMP) which is itself a variation of matching pursuit. In matching pursuit, the feature that is most correlated with the residual of the regression target, given the already chosen features, is greedily selected at each step. The OMP variant adds a recomputation of the residuals as the orthogonal projection of the target over the space of selected features. According to [21] OMP can be applied to RL as well giving the example of a Bellman residual minimization one (OMP-BRM).

In OMP-BRM the objective is to choose the features that minimize the Bellman residual, the norm difference between a value function and the application of the Bellman expectation operator to it, in a linear approximator:

$$\underset{\omega}{\operatorname{argmin}} \left\| R + \gamma \boldsymbol{\phi}' \omega - \boldsymbol{\phi} \omega \right\|_2$$

that, if we substitute $X = \boldsymbol{\phi} - \gamma \boldsymbol{\phi}'$, can be solved as a regression problem. OMP-BRM, thus, is just a regular OMP with target R and variables X.

Painter-Wakefield and Parr [21] also presented another variation of OMP, which optimizes the selection of variables for LSTD performance, known as OMP-TD that outperformed all the previous methods even tough they did not provide any theoretical justification for the algorithm as they did for OMP-BRM, On the contrary they proved that it could never find sparse solution even if there existed and that it shares the theoretical properties of a method for feature construction [31], such that each iteration of OMP-TD tightens a bound on the approximation error of the value function.

In detail, OMP-TD works just like OMP-BRM, except that instead of recomputing the residuals using a projection over the features, it uses the linear fixed point equation of LSTD:

$$\omega = (\boldsymbol{\phi}^T \boldsymbol{\phi} - \gamma \boldsymbol{\phi}^T \boldsymbol{\phi}')^{-1} \boldsymbol{\phi}^T R$$

to find the new weights for computing them.

While these are considered wrapper methods, as they optimize for the performance of a linear model, it could be argued that they reasonably work as FS methods for different algorithms without any theoretical support.

Recursive variable selection

Another method could be the perfect example of wrapper method in the supervised case: recursive variable selection (RVS) by [22].

In RVS, a regressor from supervised learning that embeds a feature selection method is used to rank all variables and select the necessary ones for estimating the reward (where an acceptable error threshold is fixed). Then, the same procedure is applied recursively to all the selceted features. The authors suggest a residuals based method, non-dissimilar to the matching pursuit, and as ranking method the Extra-Trees regressor.

The idea of the algorithm is to find the transitive closure of the dependency graph of the features correlated with the reward, as this is the best possible lossless reduction of the original MDP. The transitive closure is necessary because state features can be correlated to each other even over different time steps.

Low-Rank factorization

Some of the most advanced FS techniques nowadays use, directly or indirectly, a low-rank factorization of the (compressed) transition matrix to reduce the number of features [8] such as in [14] and [1]. Some of them even digress from FS into feature construction.

As example for all of them we will use the one to which all of the previous reduce: fast feature selection (FFS) [1].

FFS uses Singular Value Decomposition (SVD) for fast low-rank approximation of $\hat{P}^\pi$:

$$SVD(\hat{P}^\pi) = U\Sigma V^T$$

where $\hat{P}^\pi$ is computed from data through a least-square regression:

$$\hat{P}^\pi = \boldsymbol{\phi}^\dagger \boldsymbol{\phi}'$$

using Moore-Penrose pseudo inverse $\boldsymbol{\phi}^\dagger$ to solve $\boldsymbol{\phi}' = \hat{P}^\pi \boldsymbol{\phi}$ for $\hat{P}^\pi$.

The Σ matrix computed through SVD is a diagonal matrix where each element can be interpreted as the importance of the feature to predict the next state. One can thus select the k columns of U which have the corresponding k-th highest value in Σ. The selected features are then the columns of $\boldsymbol{\phi}\hat{\boldsymbol{\phi}}$, that is the new matrix of states, where $\hat{\boldsymbol{\phi}} = [U_k, \hat{R}^\pi]$ with $\hat{R}^\pi = \boldsymbol{\phi}^\dagger R^\pi$. FFS, however, builds the the final feature from a linear combination of the selected ones.

FFS is meant for batch RL, notably for LSTD, as it tries to minimize the Bellman error in the linear approximation case. It is an improvement of the basic SVD algorithm [8] and it is also theoretically the same as linear feature discovery [14], but with better performances and stronger theoretical background.

In fact, a bound on the L_2 norm of the Bellman error holds for FFS:

$$\|\mathrm{BE}_{\boldsymbol{\phi}}\|_2 \leq \|\Delta_R^\pi\|_2 + \gamma \|\Delta_{\boldsymbol{\phi}}^\pi\|_2 \|\omega_{\boldsymbol{\phi}}^\pi\|_2$$

where the error of estimating $\hat{P}^\pi$ and the error from removing the features add up:

$$\begin{aligned}
\|\Delta^\pi_\phi\|_2 &\leq \left\|\phi' - \phi\hat{P}^\pi\right\|_2 \left\|\hat{\phi}\right\|_2 + \left\|\left(I - \hat{\phi}\hat{\phi}^\dagger\right)\hat{P}^\pi\hat{\phi}\right\|_2 \|\phi\|_2 \\
\|\Delta^\pi_R\|_2 &\leq \left\|R - \phi\hat{R}\right\|_2 + \left\|\left(I - \phi\hat{\phi}^\dagger\right)^\dagger \phi\hat{R}\right\|_2
\end{aligned}$$

while ω^π_ϕ is problem and implementation dependent and available at run time.

3.4 Filter Methods

At last we review the approaches which measure the relevance and redundancy of variables through statistical measures with a clear theoretical interpretation.

Dynamic Bayesian Network

Some filter methods like [27] have ideal theoretical properties but make unrealistic assumptions such as the availability of the model of a *factored MDP*, a compact representation suitable for MDP with some regularities, as a *dynamic bayesian network* (DBN). The DBN, which includes a dependency graph, is then used to find the closure of the dependency graph correlated with the reward, just like in [22].

While there are ways to learn the model as a DBN, learning with all the features would defeat the purpose of feature selection altogether.

Features correlation

The most basic kind of filter techniques use a simple correlation statistic between state features, with no target [13]. The most popular correlation statisics are information theoretical ones: mutual information and some weighted or normalized variants plus the measure of the chi-square independency test.

In [13], the authors found that aiming at the lowest dependency in the features gives the best results as it diversifies the feature space, but this kind of FS gives no meaningful guarantees.

Reward correlation

On the other hand, in [28] correlation with the reward was considered. More specifically, a subset of features is chosen in a way that it is statistically independent of the next step reward given the complete set of features. However, in MDPs, there can be implicit delayed dependencies that this technique does not take into account.

The authors of [23] try to improve on the previous result evaluating the conditional mutual information between features at a time step k and the expected cumulative reward from that step onward J_k^π given the time step, in order to take in account delayed dependencies. Hence, they propose an algorithm that greedily minimizes for the average of $\mathrm{I}\left(J_k^\pi; \boldsymbol{\phi}_S^k | \boldsymbol{\phi}^k\right)$ over the time steps.

While there is no theory behind it that gives any guarantee on the error, this method seems sound and the most general and complete FS filter method.

Chapter 4

Theoretical Results

Here we present our main contribution: inspired by [2], we provide a bound on the control error in the RL framework such that a greedy algorithm could be developed as a filter feature selection method.

4.1 Value function bounds

In RL, the objective is to learn optimal control of MDPs, hence maximize the expected reward G_t through the choice of a policy given the initial condition. This concept is summarized by the state-value function as shown in eq. (2.2.10). As the removal of a feature could distort the values of the V-function, a feature selection method should remove only the features for which such difference is zero or, more in general, should minimize it.

The exact measure of this error is unfeasible to compute, nor to minimize: it would be necessary to evaluate both the functions for the optimal policy, on the complete set and on the subset of features, on all the state space, which could be infinite or even continuous, and then minimize the difference. A task that seems at least comparable to solving the RL problem even if the optimal policy were given, surely not fit as a filter method.

For this reasons it is preferable to minimize an upper bound on the error, which can be chosen arbitrarily expensive to compute: a trade off is possible between performance and tightness of the bound. Moreover, as the optimal policy is not available before learning, a filter method has to optimize for policy evalutaion where the policy has to be choosen wisely.

If we want to take into account actions as well as state variables as features to select, in the broad general case when also some actions could be useless, then instead of the state-value function we shall consider the action-value function an develop an upper bound for it. We proceed considering only the latter as it can always be reduced to the former by eq. (2.2.9) and all proofs can be adapted for it.

So we want to define and bound an error on the Q-function when it works on a subset of features. As already done in [2], [20] we rely on the filter assumption.

Assumption 4.1.1 (Filter)**.** *We define $\hat{Q}^\pi$ as the best possible estimator of the true Q^π value given a subset of features $\boldsymbol{\phi}_S$ instead of $\boldsymbol{\phi}$.*

This is necessary in order to optimize for the best subset of features independently of the estimator used.

Therefore we have to find a bound on the the estimator error under the above assumption:

$$\left| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right| \leq \epsilon, \tag{4.1.1}$$

where $\epsilon \geq 0$ is arbitrarily small.

One approach to the upper bound is the decomposition of the error in the *reward error*, the error in estimating the reward function $\mathcal{R}$, and the *model error*, the error in estimating the transition probabilities $\mathcal{P}$. An adaptation of the idea of *Bellman error* found in [29] to our case, using the Bellman Equations with matrix notation, yields:

$$\begin{aligned} Q^\pi - \hat{Q}^\pi &= R + \gamma P Q^\pi - \left(\hat{R} + \gamma \hat{P}\hat{Q}^\pi\right) \\ &= \left(R - \hat{R}\right) + \gamma \left(PQ^\pi - \hat{P}\hat{Q}^\pi\right) \\ &= \left(R - \hat{R}\right) + \gamma P \left(Q^\pi - \hat{Q}^\pi\right) + \gamma \hat{Q}^\pi \left(P - \hat{P}\right). \end{aligned}$$

Here we can take advantge of the recursion and solve through inversion for the Q-function difference:

$$\left(Q^\pi - \hat{Q}^\pi\right)(I - \gamma P) = \left(R - \hat{R}\right) + \gamma \hat{Q}^\pi \left(P - \hat{P}\right)$$

Now, the next typical step in RL would be to use the infinity norm and the triangle inequality:

$$\left\| Q^\pi - \hat{Q}^\pi \right\|_\infty \leq \frac{1}{1-\gamma} \left(\left\| R - \hat{R} \right\|_\infty + \gamma \|\hat{Q}^\pi\|_\infty \left\| P - \hat{P} \right\|_\infty \right) \tag{4.1.2}$$

however the error bound in L_∞ norms is expressed in terms of the uniform approximation error over the whole state space, hence it is difficult to guarantee for large discrete or continuous state spaces [16].

A slight improvement is possible given some knowledge of the d^π_μ of a MDP (hence, only in policy evaluation) as follows:

$$\left\| Q^\pi - \hat{Q}^\pi \right\|_{\rho, d^\pi_\mu} \leq \left\| R - \hat{R} \right\|_{\rho, d^\pi_\mu} + \gamma \left\| P - \hat{P} \right\|_{\rho, d^\pi_\mu} \|\hat{Q}^\pi\|_{\rho, d^\pi_\mu} \tag{4.1.3}$$

where ρ is any positive integer (or infinity), which means any norm would work. Note that the bound is on the expected value of the error under the state visitation measure instead of the maximum possible error.

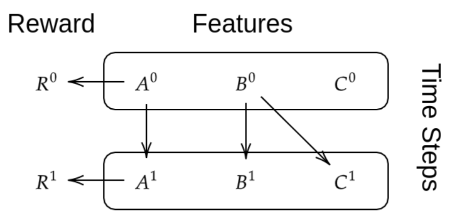

Figure 4.1: Illustration of dependencies for example 4.1

Yet, this decomposition is useless towards the development of a generalized filter method because it is not suited as a target to be minimized by a greedy algorithm.

A filter method cannot evaluate all possible subsets of features as the number is exponential in the number of features (2^n), hence an approximation must be used that is less computationally demanding. A greedy algorithm is the usual choice: each feature is evaluated only once and selected to minimize (or maximize) an objective.

This bound cannot be greedily minimized in a reliable way because the model error embeds a recursion in itself: the selected feature should minimize the error in predicting themselves at the next step, but there could be features that are useless in predicting the reward but necessary to predict themselves. It is more easily explained with an example:

Example 4.1. *Assume there are three features A,B,C where A is perfectly correlated with the reward and next(A) (A at the next step), B is useless for th reward, but is correlated with next(B) and next(C) and finally C is useless.*

A greedy algorithm that minimizes bound in eq. (4.1.3) *or eq.* (4.1.2) *would find that removing B increases the error because it would lead to a worse prediction of C at the next step even though C is actually useless and would be removed at the next step. This means that if the algorithm removes B then error would increase but when it will remove C error would decrease as it is not predicted anymore.*

As the example shows, the error does not monotonically increase with the removal of features and thusm when the algorithm stops without removing all the features, there is no guarantee that the error won't decrease by removing more of them. Clearly this is not an acceptable property in a greedy algorithm.

The example above shows the main difference with respect to the supervised learning case in the filter feature selection context: the indirect dependency of the variables with respect to the reward in a time step and consequently to the expected cumulative reward.

In order to solve this problem we should select only the features that the reward depends on and the features necessary to predict the next state of those for all possible time steps and so on recursively. Stated in a formal way: one must select all and only the neighbours features of reward in the transitive closure of the dependency graph of all the features [22].

A upper bound on the control error could be derived from the dependency graph decorating it with propagation of errors terms, independently of how the difference $\delta(\phi, \phi_S)$ between the initial conditions is computed. However this error propagation term, C_R and C_{P^π} as respectively dependent on R and P^π in the general case, even in the trivial case of constants such as Lipschitz ones, compound until the goal function of eq. (2.2.3) diverges and consequently the Q-function as well. An example formulation could be:

$$\left|\hat{Q}^\pi(\phi) - Q^\pi(\phi_S)\right| \leq C_R \sum_{k=0}^{\infty} \gamma^k C_{P^\pi} \delta(\phi, \phi_S). \tag{4.1.4}$$

The bound would start as a n-step prediction error, of which we would take the limit as the number of step n approaches infinity. Such upper bounds are are reducible to model-based bounds known to diverge except under very strong assumptions [7].

4.2 Bound on Regression

In this thesis, we use a supervised learning approach to RL. From this point of view, the evaluation of a value function, for a given policy π, is just a regression problem, where the *mean square error* (MSE) is:

$$MSE = \mathop{\mathbb{E}}_{J_t^\pi, \phi_t}\left[\left(J_t^\pi - g(\phi_t)\right)^2\right] \tag{4.2.1}$$

For such error an information theoretic bound for feature selection can be derived using [2]:

$$\inf_{g \in \mathcal{G}_S} \mathop{\mathbb{E}}_{J^\pi, \phi_S}\left[\left(J^\pi - g(\phi_S)\right)^2\right] \leq \sigma^2 + 2\|J^\pi\|_\infty^2 \mathrm{I}\left(J^\pi; \phi_{\bar{S}} | \phi_S\right), \tag{4.2.2}$$

where we omit t as subscript when it is equal to zero (and features are drawn from μ), S is the set of selected features , $\bar{S}$ is the complement set of discarded features, $\mathcal{G}_S$ is the space of all possible functions over S and σ^2 is the irreducible error in regression $\mathbb{E}_\phi\left[\left(J^\pi - \mathbb{E}\left[J^\pi|\phi\right]\right)^2\right]$. It must be noted that this inequality uses a statistical measure that is very similar to the one used for feature selection in [23] who made an average of the CMI over all time steps.

Here we need to specify an assumption under which we will operate throughout this thesis:

Assumption 4.2.1. *The whole set of features represents exactly the problem with zero error. We say that the initial (complete) set of features embeds a MDP.*

Only under this hypothesis we can guarantee that it would be possible for the irreducible error to go to zero in the previous bound. Even though we just could ignore it as it is not influenced by feature selection.

Given the above equations, a model-free approach relying on the regression bound looks then very simple: it gives information on the mean error on predicting the total discounted reward given a subset of initial features.

However, the bound above can be made more tight and still include the irreducible error term that should disappear in our RL framework when we compare the two ideal Q-functions. Moreover, the total discounted reward J^π is a function of infinite random variables R_t properly weighted. As such, it can be hard to actually evaluate and estimate its distribution.

4.3 Q-function Series Bound

We improve the previous bound by specialising it to the Q-function as a series of regression problems.

Theorem 4.3.1 (Series Bound). *Given a subset of features S and a policy π, an upper bound holds over the error introduced in the Q-function, under the filter assumption (4.1.1), such that:*

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sqrt{2} \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \sqrt{\mathrm{I}\left(R_t; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)} \tag{4.3.1}$$

where $\rho \leq 2$ and d can be any distrution over the features.

It must be noted that this bounds holds only in policy evalutaion as it depends on π.

Proof

Starting from its definition 2.2.5 as conditional expected value of the the total discounted reward given the initial features and the using th filter assumption (4.1.1) for $\hat{Q}^\pi$, we let:

$$Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) = \mathbb{E}\left[J^\pi | \boldsymbol{\phi}\right] - \mathbb{E}\left[J^\pi | \boldsymbol{\phi}_S\right], \tag{4.3.2}$$

applying the definition of J^π (see eq. (2.2.2)) we get:

$$Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) = \mathbb{E}\left[\textstyle\sum_{t=0}^{\infty} \gamma^t R_t | \boldsymbol{\phi}\right] - \mathbb{E}\left[\textstyle\sum_{t=0}^{\infty} \gamma^t R_t | \boldsymbol{\phi}_S\right].$$

Now we can use the linearity of $\mathbb{E}$ to decompose the Q-functions into the discounted sum over all the times steps:

$$Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) = \sum_{t=0}^{\infty} \gamma^t \big(\mathbb{E}\left[R_t | \boldsymbol{\phi}\right] - \mathbb{E}\left[R_t | \boldsymbol{\phi}_S\right] \big), \tag{4.3.3}$$

then apply an expected norm of any order ρ and over any distribution d whose domain are the features:

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} = \left\| \sum_{t=0}^{\infty} \gamma^t \left(\mathbb{E}\left[R_t|\boldsymbol{\phi}\right] - \mathbb{E}\left[R_t|\boldsymbol{\phi}_S\right] \right). \right\|_{\rho,d}$$

In order to move on, we need to loosen the equation using the triangle inequality for the norm:

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sum_{t=0}^{\infty} \gamma^t \|\mathbb{E}\left[R_t|\boldsymbol{\phi}\right] - \mathbb{E}\left[R_t|\boldsymbol{\phi}_S\right]\|_{\rho,d}. \tag{4.3.4}$$

As integral, without loss of generality:

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sum_{t=0}^{\infty} \gamma^t \left\| \int R_t \left(p(\,\cdot\,|\boldsymbol{\phi}) - p(\,\cdot\,|\boldsymbol{\phi}_S) \right) \mathrm{d}R_t \right\|_{\rho,d}$$

we can apply Holder inequality under the assumption that for all t there is a $\|R_t\|_\infty$ larger than all the other R_t almost surely (probabilistically speaking: always larger except for a set of measure zero):

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \left\| \int |p(\,\cdot\,|\boldsymbol{\phi}) - p(\,\cdot\,|\boldsymbol{\phi}_S)| \mathrm{d}R_t \right\|_{\rho,d}$$

The integral of an absolute difference of densities is the definition of Total Variation (see definition 2.4.4):

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq 2 \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \|D_{\mathrm{TV}} \left(p(\,\cdot\,|\boldsymbol{\phi}), p(\,\cdot\,|\boldsymbol{\phi}_S) \right)\|_{\rho,d}$$

to which we can apply the Pinsker's inequality (see definition 2.4.5):

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sqrt{2} \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \left\| D_{\mathrm{KL}}^{\frac{1}{2}} \left(p(\,\cdot\,|\boldsymbol{\phi}), p(\,\cdot\,|\boldsymbol{\phi}_S) \right) \right\|_{\rho,d}$$

Now we can explicit the expected norm in order to apply the inverse Jensen inequality for which the expected value of a concave function is smaller than the function of the expected value:

$$\begin{aligned} &\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \\ &\qquad \leq \sqrt{2} \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \left(\mathop{\mathbb{E}}_{\boldsymbol{\phi}\sim d} \left[|D_{\mathrm{KL}}\big(p(\,\cdot\,|\boldsymbol{\phi}), p(\,\cdot\,|\boldsymbol{\phi}_S)\big)|^{\frac{\rho}{2}} \right] \right)^{\frac{1}{\rho}} \\ &\qquad \leq \sqrt{2} \sum_{t=0}^{\infty} \gamma^t \|R_t\|_\infty \sqrt{ \mathop{\mathbb{E}}_{\boldsymbol{\phi}\sim d} \left[D_{\mathrm{KL}}\big(p(\,\cdot\,|\boldsymbol{\phi}), p(\,\cdot\,|\boldsymbol{\phi}_S)\big) \right] } \end{aligned}$$

It must be noted that the function of which the expected value is taken is concave only if $\rho/2 \leq 1$, so we from now on operate under then assumption that $\rho \leq 2$.

Now, the application of the definition of the conditional mutual information (definition 2.4.7) leads to our bound:

$$\left\| Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sqrt{2} \sum_{t=0}^{\infty} \gamma^t \|R_t\|_{\infty} \sqrt{\mathrm{I}\left(R_t; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)} \tag{4.3.5}$$

However this upper bound cannot be evaluated as it is an infinite sum. Later we will propose some solutions to this issue.

Interestingly, one can follow the derivation from eq. (4.3.4) but starting from eq. (4.3.2) to prove:

$$\left\| Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S) \right\|_{\rho,\mu} \leq \sqrt{2} \|J^{\pi}\|_{\infty} \sqrt{\mathrm{I}\left(J^{\pi}; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)}, \tag{4.3.6}$$

with $\rho \leq 2$, which is by itself a generalization of the regression bound in eq. (4.2.2).

4.4 Cutting the Tail

The Series Bound (theorem 4.3.1), while it gives interesting insights, is useless from a practical point of view as its evaluation need an infinite number of estimation of CMI. Here we explore some of the possible ways of limiting the sum to a finite number of terms.

Discrete Bound

Corollary 4.4.0.1 (Discrete Bound)**.** *When both features and reward are discrete the follows applies:*

$$\left\| Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sqrt{2} \Big(\sum_{t \in T} \gamma^t \|R_t\|_{\infty} \sqrt{\mathrm{I}\left(R_t; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)} + \sqrt{\mathrm{H}\left(\boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)} R_{max} \sum_{t \in \bar{T}} \gamma^t \Big) \tag{4.4.1}$$

where T is the finite set of selected time steps, $\bar{T}$ is the complementary set and we let $R_{max} = \max_t |R_t|$.

Proof. In the discrete case, where entropy is non-negative, we can state a bound on the conditional mutual information:

$$\begin{aligned} \mathrm{I}\left(R_t; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right) &= \mathrm{I}\left(R_t; \boldsymbol{\phi}\right) - \mathrm{I}\left(R_t; \boldsymbol{\phi}_S\right) \\ \textit{(MI definition)} &= \mathrm{H}\left(\boldsymbol{\phi}\right) - \mathrm{H}\left(R_t, \boldsymbol{\phi}\right) - \mathrm{H}\left(\boldsymbol{\phi}_S\right) + \mathrm{H}\left(R_t, \boldsymbol{\phi}_S\right) \\ \textit{(As cond. entropy)} &= \mathrm{H}\left(\boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right) - \mathrm{H}\left(\boldsymbol{\phi}_{\bar{S}} | R_t, \boldsymbol{\phi}_S\right) \\ \textit{(Non-negativity)} &\leq \mathrm{H}\left(\boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right), \end{aligned}$$

hence we can use it to bound an arbitrarily number of time steps of the sum in theorem 4.3.1 obtaining the above result. QED

Continuous Bound

In the continuous case, where the differential entropy can be negative the former bound does not hold and a general upper bound to the mutual information (conditional or otherwise) is challenging [5], but it is possible to bound the total variation distance.

Corollary 4.4.0.2 (Continuous Bound)**.** *In a worst case scenario thus the following can be used:*

$$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,d} \leq \sqrt{2}\Bigg(\sum_{t\in T}\gamma^t\|R_t\|_\infty\sqrt{\mathrm{I}\left(R_t;\boldsymbol{\phi}_{\bar{S}}|\boldsymbol{\phi}_S\right)} + \sqrt{2}R_{max}\sum_{t\in\bar{T}}\gamma^t\Bigg)$$

which is identical to the discrete case but with constant entropy equal to 2*.*

Proof. It follows straightforwardly from theorem 4.3.1 by bounding to 1 the total variation in one of the steps of the proof:

$$\begin{aligned}\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,d} &\leq 2\sum_{t=0}^{\infty}\gamma^t\|R_t\|_\infty\|D_{\mathrm{TV}}\left(p(\,\cdot\,|\boldsymbol{\phi}), p(\,\cdot\,|\boldsymbol{\phi}_S)\right)\|_{\rho,d}\\ \textit{(}D_{\mathrm{TV}}\leq 1\textit{)} &\leq \sqrt{2}\sum_{t=0}^{\infty}\gamma^t\|R_t\|_\infty\sqrt{2}. \end{aligned} \tag{4.4.2}$$

QED

Choice of time steps

The intuitive choice, in the last two bounds (corollaries 4.4.0.1 and 4.4.0.2), for the set of selected time steps T is to limit it to the first k. Then we have:

$$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,d} \leq \sqrt{2}R_{max}\Bigg(\sum_{t=0}^{n}\gamma^t\sqrt{\mathrm{I}\left(R_t;\boldsymbol{\phi}_{\bar{S}}|\boldsymbol{\phi}_S\right)} + \frac{\gamma^{k+1}}{1-\gamma}\sqrt{\mathrm{H}\left(\boldsymbol{\phi}_{\bar{S}}|\boldsymbol{\phi}_S\right)}\Bigg), \tag{4.4.3}$$

and similarly:

$$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,d} \leq \sqrt{2}R_{max}\left(\sum_{t=0}^{n}\gamma^t\sqrt{\mathrm{I}\left(R_t;\boldsymbol{\phi}_{\bar{S}}|\boldsymbol{\phi}_S\right)}+ \right.$$
$$\left. + \frac{\gamma^{k+1}}{1-\gamma}\sqrt{2}\right) \quad (4.4.4)$$

Bellman Bound

A solution that works in both the continuous and the discrete cases and does not introduce a constant error term is possible as well.

Corollary 4.4.0.3 (Bellman Bound)**.** *Without adding any assumption, a bound that is zero when S includes all the features is:*

$$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,\mu} \leq \sqrt{2}\left(\sum_{t=0}^{k-1}\gamma^t\|R_t\|_\infty\sqrt{\mathrm{I}\left(R_t;\boldsymbol{\phi}_{\bar{S}}|\boldsymbol{\phi}_S\right)} + \right.$$
$$\left. + \gamma^k\|J_k^\pi\|_\infty\sqrt{\mathrm{I}\left(J_k^\pi;\boldsymbol{\phi}_{\bar{S}}^k|\boldsymbol{\phi}_S^k\right)}\right) \quad (4.4.5)$$

where $\rho \leq 2$.

Proof. We can exploit the recursive property of $J_t^\pi = \gamma J_{t+1}^\pi$ in eq. (4.3.2):

$$Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) = \mathbb{E}\left[\textstyle\sum_{t=0}^{k-1}\gamma^t R_t + \gamma^k J_k^\pi|\boldsymbol{\phi}\right]$$
$$- \mathbb{E}\left[\textstyle\sum_{t=0}^{k-1}\gamma^t R_t + \gamma^k J_k^\pi|\boldsymbol{\phi}_S\right].$$

As before, by applying the linearity of the expectation:

$$Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) = \sum_{t=0}^{k-1}\gamma^t\big(\mathbb{E}\left[R_t|\boldsymbol{\phi}\right] - \mathbb{E}\left[R_t|\boldsymbol{\phi}_S\right]\big) +$$
$$+ \gamma^k\big(\mathbb{E}\left[J_k^\pi|\boldsymbol{\phi}\right] - \mathbb{E}\left[J_k^\pi|\boldsymbol{\phi}_S\right]\big)$$

This is a recursive definition: in the right hand side the first part is akin to eq. (4.3.3) but on finite time steps while the second part is the difference of the two Q-function as the left hand side, just k time steps forward.

One then only needs to use the triangle inequality in order to separate the two parts:

$$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{\rho,d} = \sum_{t=0}^{k-1}\gamma^t\|\mathbb{E}\left[R_t|\boldsymbol{\phi}\right] - \mathbb{E}\left[R_t|\boldsymbol{\phi}_S\right]\|_{\rho,d} +$$
$$+ \gamma^k\left\|Q^\pi(\boldsymbol{\phi}^k) - \hat{Q}^\pi(\boldsymbol{\phi}_S^k)\right\|_{\rho,d'}$$

the first clearly leads to eq. (4.3.5) on finite time steps while the second can be bounded by eq. (4.3.6) where the wighting distribution is $d' = P^k d$:

$$\left\| Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S) \right\|_{\rho,d} \leq \sqrt{2} \left(\sum_{t=0}^{k-1} \gamma^t \|R_t\|_\infty \sqrt{\mathrm{I}\left(R_t; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)} + \right. \\ \left. + \gamma^k \|J_k^\pi\|_\infty \sqrt{\mathrm{I}\left(J_k^\pi; \boldsymbol{\phi}_{\bar{S}}^k | \boldsymbol{\phi}_S^k\right)} \right)$$

If we let $d = \mu$, then $d' = P^k \mu$ is just the distribution of features at time step k and we get the result. Consequently, $\boldsymbol{\phi}^k$ are the features at the k time step. QED

It must be noted that this bound gets closer to eq. (4.3.6) as k gets smaller, and coincides when k is zero. The hyper-parameter k can then be seen as a trade-off between accuracy and the number of required mutual information computations (strictly related to performance).

All these proofs, except for the last one, hold for any $\rho \leq 2$ and d as distributions of $\boldsymbol{\phi}$, however we focus on the μ distribution of the initial states as then the bounded quantity has the meaning of mean loss in reward with less features in a complete episode. This is also consitent with corollary 4.4.0.3.

Chapter 5

Greedy Algorithms

All of the previous finite bounds corollaries 4.4.0.1 to 4.4.0.3 can be evaluated and as such can be used to develop greedy algorithms with a theoretically meaningful stopping condition.

Greedy algortihms have been a staple of feature selection in supervised learning as they are one of the simplest search approaches that can find a local optimum for the subset selection problem in polynomial time. Notable examples are [2], [20] where such greedy algorithms on similar bounds are shown to be optimal as long as the conditional mutual information is zero.

5.1 Algorithms

We propose then greedy algorithms for both forward and backward feature selection: respectively algorithms 1 and 2, that could operate with any of the above mentioned bounds that differ only for the corrective term, but as an example are shown with the last one (corollary 4.4.0.3).

The idea of the BackwardFS (ForwardFS) algorithm is to start from an initial full (empty) set of selected features then for each feature i evaluate the bound at $S = S \setminus \{i\}$ ($S = S \cup \{i\}$) and remove (keep) the feature for which the bound is minimum. The procedure is repeated with the new set S until the stopping condition is met: the value of the bound is larger then the acceptable error on the Q-function which is an input.

In the forward case the minimization of the bound is done indirectly as maximization the second term of the chain rule decomposition of CMI as:

$$\begin{aligned} \underset{S}{\operatorname{argmin}} \, \mathrm{I}\left(\cdot\, ; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S \right) &= \underset{S}{\operatorname{argmin}} \, \mathrm{I}\left(\cdot\, ; \boldsymbol{\phi} \right) - \mathrm{I}\left(\cdot\, ; \boldsymbol{\phi}_S \right) \\ &= \underset{S}{\operatorname{argmax}} \, \mathrm{I}\left(\cdot\, ; \boldsymbol{\phi}_S \right) \end{aligned}$$

5.2 Chain Rule

It must be noted that, especially for the bounds eqs. (4.4.4) and (4.4.5), the chain rule of the conditional mutual information eq. (2.4.12) can be used to improve both accuracy and performance of the evaluation of the bound (with regards to the complexity of the estimation of the CMI in high dimensionality).

Here we derive how to compute $\mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right)$ as sum of CMIs of the previous step of the backward algorithm (in the backward case), similarly to [2]:

$$\begin{aligned}
\mathrm{CMI}^j &= \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}} | \boldsymbol{\phi}_S\right) \\
&= \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}\right) - \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_S\right) \\
&= \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}\setminus\{i\}}, \boldsymbol{\phi}_{S\cup\{i\}}\right) - \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_S\right) \\
&= \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}\setminus\{i\}} | \boldsymbol{\phi}_{S\cup\{i\}}\right) + \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{S\cup\{i\}}\right) - \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_S\right) \\
&= \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}\setminus\{i\}} | \boldsymbol{\phi}_{S\cup\{i\}}\right) + \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_i | \boldsymbol{\phi}_S\right) \\
&= \mathrm{CMI}^{j-1} + \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_i | \boldsymbol{\phi}_S\right)
\end{aligned}$$

leading to:

$$\mathrm{CMI}^j = \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{\bar{S}^j} | \boldsymbol{\phi}_{S^j}\right) = \sum_{c=0}^{|\bar{S}|} \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}_{i^c} | \boldsymbol{\phi}_{S^c}\right) \tag{5.2.1}$$

where S^j is the set of selected features at the time step j of the algorithm and likewise for CMI^j and $\bar{S}^j$. It works analogously for the forward case. It is implied that the base step of the recurrence relation is:

$$\mathrm{CMI}^0 = \mathrm{I}\left(\,\cdot\,; \emptyset | \boldsymbol{\phi}\right) = 0$$

while an eventual final step would compute:

$$\mathrm{CMI}^{|\boldsymbol{\phi}|} = \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi} | \emptyset\right) = \mathrm{I}\left(\,\cdot\,; \boldsymbol{\phi}\right)$$

Remark. The computation of this step by the greedy algorithm would mean that it is discarding all the features, unreasonable in practice. However, from a theoretical point of view, it is a general upper bound on the mean possible difference between two Q-function in policy evaluation. This is itself upper bounded by the well known infinity norm bound per eq. (4.4.2):

$$\begin{aligned}
\left\| Q^\pi - \hat{Q}^\pi \right\|_{2,\mu} &\leq \sqrt{2} \sum_{t=0}^{T-1} \gamma^t \|R_t\|_\infty \sqrt{\mathrm{I}\left(R_t; \boldsymbol{\phi}\right)} + \gamma^T \|J\pi_k\|_\infty \mathrm{I}\left(R_t; \boldsymbol{\phi}\right) \\
&\leq \frac{2R_{max}}{1-\gamma}
\end{aligned}$$

An analogous result holds true for the forward selection:

$$\begin{aligned}\mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_S\right) &= \mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_{S\setminus\{i\},\boldsymbol{\phi}_i}\right)\\ &= \mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_i|\boldsymbol{\phi}_{S\setminus\{i\}}\right) + \mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_{S\setminus\{i\}}\right),\end{aligned}$$

that, combined with the base case $\mathrm{I}\left(\,\cdot\,;\emptyset\right) = 0$, means:

$$\mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_S\right) = \sum_{c=0}^{|S|} \mathrm{I}\left(\,\cdot\,;\boldsymbol{\phi}_{i^c}|\boldsymbol{\phi}_{S^c}\right).$$

5.3 Computational Complexity

This algorithms, as greedy ones, have a reasonably low complexity and should be fairly efficient.

A rigorous analysis of its time complexity is hard as it depends on the chosen threshold ϵ. However, it is obvious that the main bottleneck in the algorithms is the computation of the conditional mutual information. which by itself depends on the input and the chosen estimator, but the analysis of the complexity of those is out of the scope of this thesis.

For the sake of simplicity we will assume the time complexity of the estimator to be $O(n \log n)$ where n is the number of trajectories considered, this ignores the impact of dimensionality on the estimator which is supposed to scale reasonably with its increase. The number of time the CMI is estimated depends on the number of features m, on the number of time steps considered k and on the ratio of features that are actually discarded x (the ratio of selected features is $(1-x)$).

In order to fill the matrix, containing $k \times m$ cells, one time for each of the $x \cdot m$ steps of the whole algorithm we calculate the CMI in asymptotically $O(kxm^2)$ steps.

In more detail, for BackwardFS the CMI is computed:

$$k\frac{(xm+1)(m+(1-x)m)}{2} = k\frac{m^2x(1-x)+m(2-x)}{2}$$

times, as the number of features in the column of the matrix decreases by one at each step, but the asymptotical complexity is the same. The forward case is analogous.

Algorithm 1 Greedy Backward Feature Selection

```
1: Input
2:     $T$    Trajectories long at least $k$ time steps
3:     $k$    Number of time steps to consider
4:     $\gamma$    Discount factor
5:     $\epsilon$    Error threshold
6: Output
7:     $S$    The set of selected features
8: procedure BackwardFS($T, k, \gamma, \epsilon$) $\rightarrow S$
9:     $S \leftarrow \{0 \dots |\boldsymbol{\phi}|\}$    ▷ We start selecting all features
10:    $idToDiscard \leftarrow \emptyset$
11:    $err \leftarrow 0$    ▷ With zero error
12:    for $t = 0$ to $k - 1$ do
13:        $norms[t] \leftarrow \max R$ over episodes in $T$
14:    $norms[k] \leftarrow \max J_k^\pi$ over episodes in $T$
15:    for $t = 0$ to $k$ do    ▷ With $CMI^0$
16:        $\mathrm{CMI}^j[t] \leftarrow 0$
17:    while $err < \epsilon$ do    ▷ Stop when exceeding threshold
18:        $S \leftarrow S \setminus \{idToDiscard\}$
19:        for $i \in S$ do    ▷ For each features remaining
20:            for $t = 0$ to $k - 1$ do    ▷ For each time step
21:                $\mathrm{matrix}[t, i] \leftarrow \gamma^t \cdot norms[t] \cdot \mathrm{I}\left(R_t; \boldsymbol{\phi}_i | \boldsymbol{\phi}_S\right)$
22:            $\mathrm{matrix}[k, i] \leftarrow \mathrm{I}\left(J_k^\pi; \boldsymbol{\phi}_i^k | \boldsymbol{\phi}_S^k\right)$
23:        $idToDiscard \leftarrow \mathrm{argmin}_i \sum_{t=0}^k matrix[t, i]$
24:        for $t = 0$ to $k$ do
25:            $\mathrm{CMI}^j[t] \leftarrow \mathrm{CMI}^j[t] + matrix[t, idToDiscard]$
26:        $err \leftarrow \sum_{t=0}^k \sqrt{2\mathrm{CMI}^J[t]}$
27:    return $S$
```

Algorithm 2 Greedy Forward Feature Selection

```
1: Input
2:     $T$    Trajectories long at least $k$ time steps
3:     $k$    Number of time steps to consider
4:     $\gamma$    Discount factor
5:     $\epsilon$    Error threshold
6: Output
7:     $S$    The set of selected features
8: procedure ForwardFS($T, k, \gamma, \epsilon$) $\rightarrow S$
9:     $\bar{S} \leftarrow \{0 \dots |\boldsymbol{\phi}|\}$    ▷ We start discarding all features
10:    $idToAdd \leftarrow \emptyset$    ▷ Here $S$ is defined as complement of $\bar{S}$
11:    for $t = 0$ to $k-1$ do
12:        $norms[t] \leftarrow \max R$ over episodes in $T$
13:    $norms[k] \leftarrow \max J_k^\pi$ over episodes in $T$
14:    for $t = 0$ to $k-1$ do    ▷ With $CMI^0$
15:        $\mathrm{CMI}^j[t] \leftarrow \gamma^t \cdot norms[t] \cdot \mathrm{I}\left(R_t; \boldsymbol{\phi}\right)$
16:    $\mathrm{CMI}^j[k] \leftarrow \gamma^t norms[t] \mathrm{I}\left(J_k^\pi; \boldsymbol{\phi}\right)$
17:    while $err < \epsilon$ do    ▷ Stop when exceeding threshold
18:        $\bar{S} \leftarrow \bar{S} \setminus \{idToAdd\}$
19:        for $i \in \bar{S}$ do    ▷ For each features remaining
20:            for $t = 0$ to $k-1$ do    ▷ For each time step
21:                $\mathrm{matrix}[t, i] \leftarrow \gamma^t \cdot norms[t] \cdot \mathrm{I}\left(R_t; \boldsymbol{\phi}_i | \boldsymbol{\phi}_S\right)$
22:            $\mathrm{matrix}[k, i] \leftarrow \mathrm{I}\left(J_k^\pi; \boldsymbol{\phi}_i^k | \boldsymbol{\phi}_S^k\right)$
23:        $idToAdd \leftarrow \mathrm{argmax}_i \sum_{t=0}^{k} matrix[t, i]$
24:        for $t = 0$ to $k$ do
25:            $\mathrm{CMI}^j[t] \leftarrow \mathrm{CMI}^j[t] - matrix[t, idToDiscard]$
26:        $err \leftarrow \sum_{t=0}^{k} \sqrt{2\mathrm{CMI}^J[t]}$
27:    return $S$
```

Chapter 6

Experiments and Results

In this chapter we present the experiments executed to test the accuracy of the feature selection algorithms on multiple environments and to test the evaluation of the theoretical upper bound itself in comparison with the actual Q-functions.

6.1 Technical details

The actual implementation of the proposed algorithms required various lower-level decisions. Here we summarize most of them as well as our setup with the purpose of making the results reproducible. However the repository with all the code and results of the experiments is freely available [1].

Time steps sampling

The discrete and continuous variation of the upper bound eqs. (4.4.3) and (4.4.4) allow the use of any selection of the time steps to be considered after having fixed the total number of them. For this reason some methods have been designed to choose them:

1. Select one each fixed number of steps until exhaustion;
2. Random sampling with decreasing probability proportional to the distance from step zero;
3. Variance sampling, in which the steps with maximum variance are chosen based on the intuitition that those are the one with the most information.

However, after some preliminary tests, none of those gave results significantly better than the method of choosing the first k steps. In this way we also maintain consistency with the last bound eq. (4.4.5).

[1] `https://github.com/GuidoBallabio/RL-feature-selection/`

Conditional mutual information estimation

As shown in section 5.2 we use the chain rule to improve accuracy of the mutual information estimator. In this way we use the knowledge that the the total CMI increases at each step of the algorithm, relieving the estimator from this burden. Moreover, the estimator operates on a decreasing number of dimensions, and thus it is also less computationally expensive.

This has been verified after some test: this modification gave much more reliable results than the straightforward approach.

In order to estimate the CMI we used a slightly improved version of the well known KSG estimator [34] (implementation by [6]), however due to a saturation problem [9] in this estimator we contributed to the original library an implementation of a correction procedure by [15] with some minor addition proposed by [3] for mutual information and adapted by us for CMI (appendix A.4). Details in appendix A.

Q-function estimation

Unless otherwise stated, we estimate Q^π from trajectories, in order to evaluate the expected norm of the difference of the Q-function over the whole set of features and the selected subset, through an adaptation of fitted Q-iteration (FQI) for policy evaluation.

From the trajectories we prepare a database of tuples (s, a, r, s', a') where s' and a' are the state and action at the time step following s and a for all (s, a) in the trajectories, then a regressor (in our implementation Extra-Trees) is trained to predict the reward r from (s, a). Finally the regressor is trained to predict $r + \gamma\,\mathbb{E}_{(s',a'|s,a)}\left[Q^\pi(s', a')\right]$, as for the Bellman expectation equation, from (s, a) where the expectation is approximated as the iterated prediction of the regressor on (s, a). The algorithms stops after convergence given an ϵ or after a maximum number of iterations, we use the latter. Refer to the pseudo-code in algorithm 3 for details.

Trajectory generation

The algorithms require n trajectories all drawn from the same MDP with the same policy π. The sampling of features from distribution μ is executed by taking the value of the features at time step zero from the trajectories. We applied a similar procedure for sampling from ${P^\pi}^k\mu$ and for finding the max norm over R_t for each time step.

We tested also the option of considering each time step of a trajectory as the beginning of a new trajectory, thus reusing it multiple times. While this procedure was shown in some preliminary tests to achieve much higher sample efficiency with regards to the number of trajectories, it is more akin to sample from d_μ^π instead of μ and hence weaker from a theoretical point of view. As such it was discarded in favor of μ sampling.

Algorithm 3 FQI policy evaluation

```
1: Input
2:     $D$  Database of $(s, a, r, s', a')$ from trajectories with policy $\pi$
3:     $n$  Number of iterations
4:     $\gamma$  Discount factor
5:     $M$  A supervised regression algorithm with $train(x, target)$ and
       $predict(x)$ functions
6: Output
7:     $Q^\pi$  Q-function estimator as model with $predict((s, a))$
8: procedure PolicyEval($D, N, \gamma$) $\rightarrow Q^\pi$
9:     $model \leftarrow M()$
10:    for $(s, a, r, s', a') \in D$ do
11:        $model \leftarrow model.train((s, a), r)$
12:    for $i = 0$ to $n$ do
13:        $oldModel \leftarrow model$
14:        $model \leftarrow M()$
15:        for $(s, a, r, s', a') \in D$ do
16:            $target \leftarrow r + \gamma \cdot oldModel.predict((s', a'))$
17:            $model \leftarrow model.train((s, a), target)$
18:    return $model$
```

The length of the trajectories was fixed at 70, i.e., the maximum $k = 50$ used plus an arbitarily 20 for the evaluation of the corrective term of the bound in eq. (4.4.5).

Backward algorithm

The backward and forward algorithms should theoretically give the same results each time. However, due to a negative bias in mutual information estimation we incurred in when there are strongly dependent variable, the forward one was discarded in favor of the backward one. The former in fact starts minimizing the upper bound from its maximum value, which is estimated lower than the ground truth, giving thus wrong results of the bound in the first steps while the latter always begins with correct estimates eventually miscalculating at the end, which it should not ever occur in practice. All the experiments then use the backward variant (algorithm 1).

Policy

As the bound is in policy evaluation the choice of the policy on which the trajectories are based is fundamental. Intuitively we would like a policy that explores all the states and actions. Because the problem of finding the optimal exploration policy is known to be hard, we settled for toy problems with small state space, where such policy is not necessary [4] and the uniform random policy or slightly suboptimal policies could be used instead.

6.2 Environments

The approach was mainly tested using three environments: the linear quadratic gaussian (LQG) control problem, the Lunar Lander environment and a custom variant of the Taxi environment from OpenAI Gym [12].

LQG

The first one is an ideal deterministic control problem taken in a variable number of dimensions with continuous features, that we judge an ideal toy problem for robotic applications. In LQG a point in n dimensions takes a reward, quadratic, weighted in its position relative to the origin and in the magnitude of the action, the displacement for the next state. In formulas:

$$\begin{aligned} s' &= As + Ba \\ r &= -\frac{1}{2}\left(s^T Q s + a^T R a\right) \end{aligned}$$

where the state s ia a vector representing the position in the n dimension space, s' the next state, a the action as n dimensional vector and A, B, Q, R are the weight matrices.

This gives the environment some nice theorical properties, for example there are closed forms solution for the optimal policy and the Q-function [36].

The implementation used for the experiments (unless otherwise stated) has 4 dimension for 4 state features and 4 action features, of which 2 dimensions are almost useless as they have extremely small weights. Hence the features that should be eliminated are the coordinates and actions in those 2 dimensions (with subscript larger than 1).

We use this environment's characteristics to better visualize and evaluate with a feature selection method. In fact without a common stopping condition it is hard to evaluate two different feature selection methods, or the same method with different parameters. While some common ways are to use the Bellman error or an approximate value function given the selected features [1], the former is impossible without the correct model and the second one is of much less interest without a precisely defined learning method as in wrapper FS. Moreover it is computationally demanding in the non-linear case.

Thus we use LQG as a benchmark: we build an LQG environment with a weight for each dimension (with the same weight for states and actions) in decreasing order. All the dimension are independent: A, B, Q, R are diagonal and such diagonals are all equal. Then for each step of the algorithms, for each subset selected until exhaustion, we take as score the sum of the weights of the selected feature divided by the cardinality of the set. In this way it is easy to create an ideal FS method and all the sets it would select at each step for the purpose of giving an ideal baseline.

Lunar Lander

Lunar Lander is a standard problem in OpenAI Gym which tries to model the landing on a planet of a vehicle with only one discrete action. The state features are redundant as they include both position and velocity in two dimensions while only one is necessary, moreover some features, like the inclination angle, horizontal position and velocity and ground contact with feet, are less relevant as they are used only in some edge cases. Depending on the error threshold we should then remove the less relevant and the redundant while always keeping the position or velocity and of course the action.

Custom Taxi

The predefined taxi environment in OpenAI gym is a discrete toy problem where a taxi should pick up a passenger and drop it at one of four possible destinations. In the original implementation the 4 state features are compressed in one single number: the number of the state of the 500 possible ones. Of course it renders feature selection impossible in this way, which would select some states and ignore the others instead of the actual features, as such we decode (technically unravel) the features. While all the features are clearly necessary we expect the action to be fundamental and the minimum threshold to remove a feature to be high.

6.3 Policy Comparison

In order to understand the effect of the policy choice on feature selection we tested the LQG environment with a random policy (RND), the optimal deterministic policy (OPT-DET), the optimal policy with some noise (OPT-RND) and finally a gaussian sub-optimal policy with (GAU-RND) and without noise (GAU-DET).

Figure 6.1 shows that, w.r.t. to feature selection, non-deterministic policies outperform deterministic ones as they explore more states and actions but also because all the subsequent states are completely dependent on the initial one rendering the actions completely useless for estimation of J^π in a deterministic environment as LQG. Nonetheless the selection of the state features is always correct, and thus in deterministic environments the algorithms and bound can just be adapted for the V-function with optimal results.

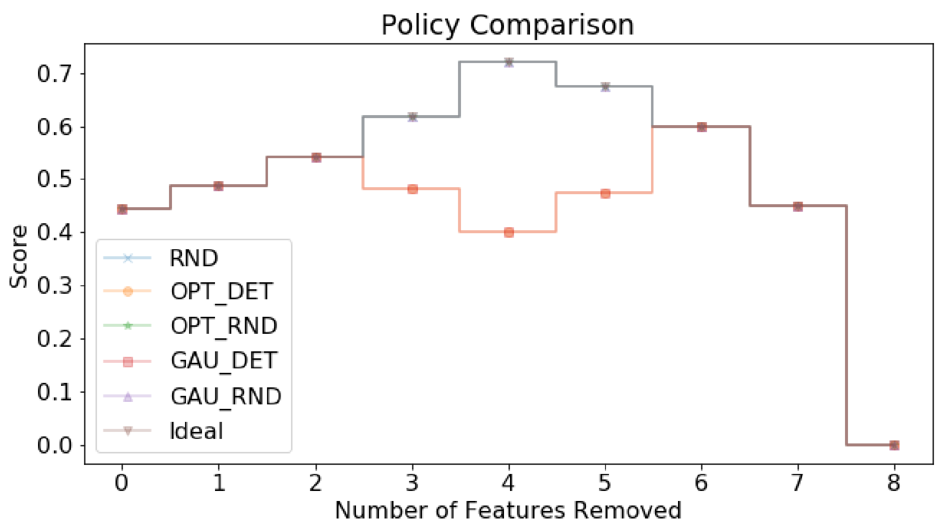

Figure 6.1: Comparison of policies

6.4 Dependency on Hyper-parameters

In order to get a sense of the most reasonable choice of the input hyper-parameters we tested the backward algorithm for three possible values of γ, four numbers of time steps k and three numbers of trajectories n, under the random policy on all three environments.

We would expect the feature selection and bound estimation to be better for increasing value of k and n, while they should get worse when we increase γ as farther away future time steps become more relevant.

The algorithm however chooses the right subset of features eventually in all experiments (the ideal one in LQG) showing robustness to these parameters. Moreover we observe that the estimation of the bound increases with γ, as it's reasonable, so an eventual error threshold should take it into account. We also note that with the less reliable estimators when we increase γ the algorithm needs a larger k, but as trajectories diverge while becoming longer, also a larger n is needed when increasing k. So the choice of γ is the most influential: dictating an increase of k and n.

For the sake of completeness, we ran some tests with $k = 0$ and unsurpisingly the results were completely random as the final discounted return while slightly dependent on the initial features cannot be correlated to them through MI estimation with current state of the art estimators at least.

The amount of dimensions and tests hinders the ability to visualize all this data synthetically and plots of the score in LQG would overlap, so we include some tables as example in appendix B.

6.5 Verification of Upper Bound Theory

About the soundness of using the error as a stopping condition, we verified the theory: that the estimated error for feature selection used as stopping condition is actually an upper bound of the estimated expected norm of the difference of Q-functions, a.k.a Q-function error.

An example of the upper-bounding is given in table 6.1 and graphically in fig. 6.1. The values show that the bound could be tighter: the maximum possible Q-function error in this case ($\rho = 2$) can be at most $\frac{R_{max}}{1-\gamma}$ which should be around 1500. The bound is very close to this order of magnitude while the actual estimated Q-function error is much smaller.

Moreover there is a limitation of the conditional mutual information estimator to take in to account: it indirectly computes the conditional probability, as a discriminative model in supervised learning, but an RL problem as a regression is extremely hard so the capacity of the kNN method used inside the estimator is insufficient, giving us just a lower bound on mutual information which theoretically has infinite capacity (it has a negative bias).

It must also be noted that we use Extra-Trees for Q-function estimation, as regressor in the policy evaluation algorithm, which is resilient to overfitting, since it uses random forests, and to useless features as well [37]. As such, with a less powerful regressor or a different policy evaluation algorithm the estimated true error could be higher. On the other hand, our bound relies on the filter assumption (4.1.1) and thus, it holds for the best possible Q-funtion estimator.

However the estimation of the value of the bound should be more reliable when less relevant features are being discarded as the CMI estimator gives better approximations for not strongly dependent variables. Thus a low value should be set as threshold for the stopping condition.

Table 6.1, as well as those in appendix B, can also be used as a reference for a possible heuristic in order to choose the input threshold. The last row show the value of the bound when no feature is selected, that is the MI between the rewards and all the features. The threshold could be chosen proportionally to that value in any environment after its estimation, and it should be at least an order of magnitude smaller.

6.6 Comparison with Fast Feature Selection

We test the soundness and performances of our feature selection method against a state of the art one, such it is FFS [1]. For this purpose we use a high-dimensional LQG environment as benchmark: with a decreasing relevance (weight) for each dimension and random features with weights equal to zero. As FFS returns linear combination of features we compute its score as the same linear combinations of the weights in absolute value, and properly normalized.

Set of Selected Features	$\left\| Q^{\pi}(\phi) - \hat{Q}^{\pi}(\phi_S) \right\|_{2,\mu}$	Bound
$\{s_0, s_1, s_2, s_3, a_0, a_1, a_2, a_3\}$	0	0
$\{s_0, s_1, s_2, a_0, a_1, a_2, a_3\}$	7.1647e-07	17.96
$\{s_0, s_1, s_2, a_0, a_1, a_2\}$	5.4779e-07	79.353
$\{s_0, s_1, a_0, a_1, a_2\}$	6.1839e-07	167.33
$\{s_0, s_1, a_0, a_1\}$	6.13e-07	213.24
$\{s_0, s_1, a_0\}$	0.28169	517.61
$\{s_0, a_0\}$	2.2457	624.28
$\{s_0\}$	58.419	737.33
$\emptyset$	97.289	800.14

Table 6.1: Verification of upper bound. Backward FS on LQG environment using bound in eq. (4.4.5), tested with $\gamma = 0.9, n = 5000, k = 40$.

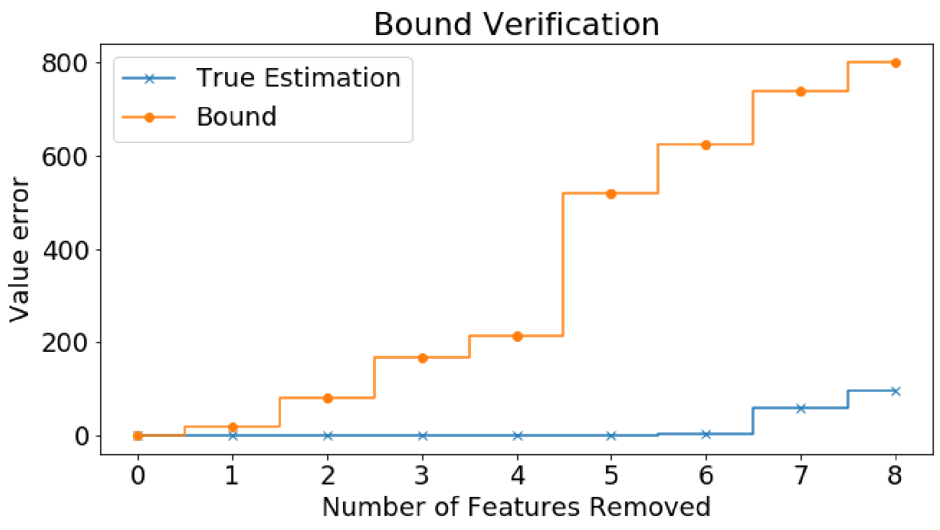

Figure 6.1: Plot of table 6.1, estimation vs. upper bound

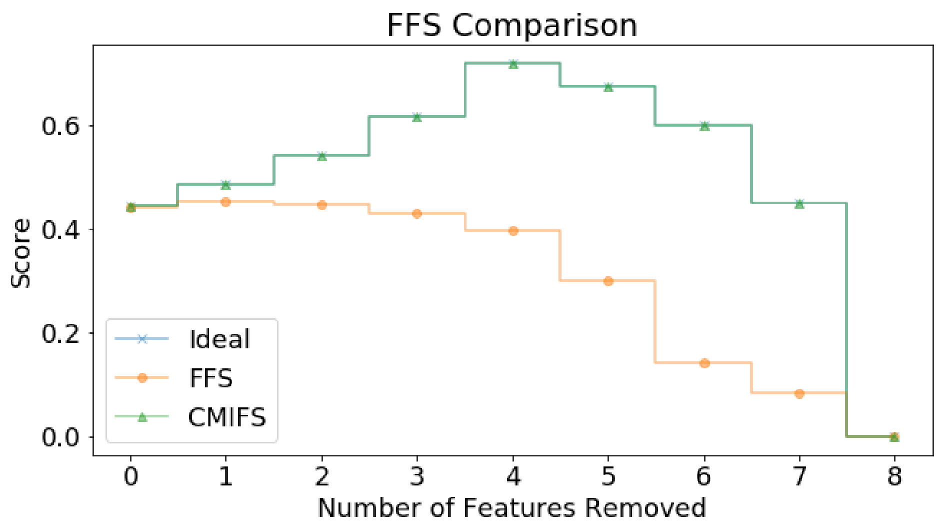

Figure 6.1: Comparison against FFS with ideal score in 4 dimensions LQG

The results of our algorithm (CMIFS) over some runs are clearly better, even optimal, in the standard example (fig. 6.1) as well as in a higher dimensional one (fig. 6.2).

However when the dimensionality gets higher the performance of our algorithm starts to deteriorate unless the CMI estimator gets as input a larger number of trajectories and a different parameter (the k in the kNN part of the MI estimator) for which a clear heuristic is not available [23]. Generally, as the idea behind the MI estimator is sample-spacing (see appendix A), increasing it gives better results but there is no guarantee about this possibility in all cases as can be seen in fig. 6.2).

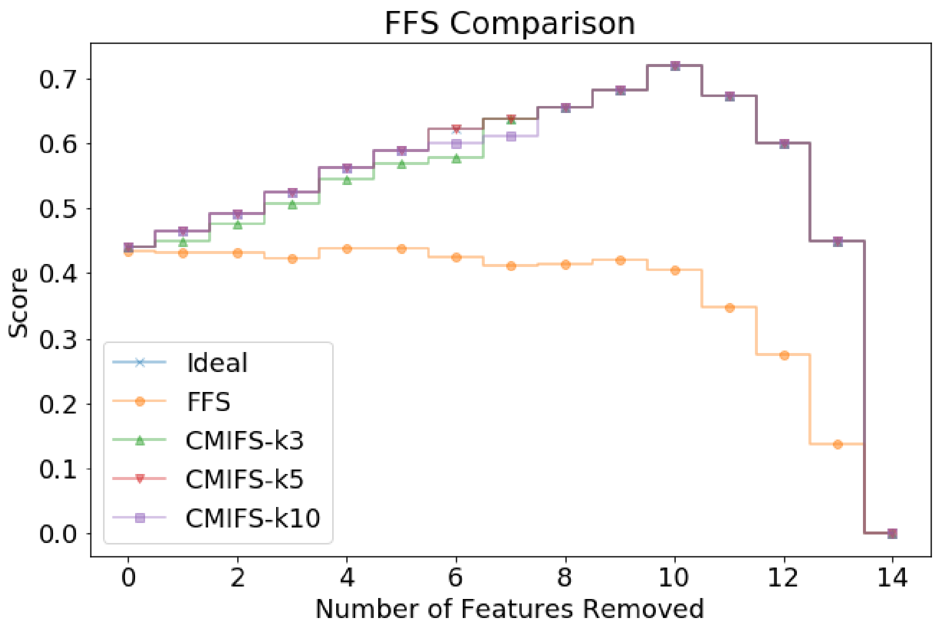

Figure 6.2: Comparison against FFS with ideal score in 7 dimensions LQG with $k = 3$, $k = 5$ and $k = 10$

Chapter 7

Conclusion

We derived an upper bound on the Q-function error given a subset of features of a MDP, then we proposed a greedy algorithm for feature selection and grounded on such theory. Finally, we experimentally verified its soundness and tested it is performance on some toy problems. In some small synthetic benchmarks it performed ideally, and much better than another feature selection algorithm (FFS [1]). However we found hard to scale it to higher dimensionalities because conditional mutual information estimators do not scale easily and proved unstable.

While we succeeded in slightly improving the estimators' performances we incurred in hard theoretical limits that could not be overcome without a completely different approach. Our insight is that most MI estimators, while they do not directly approximate the conditional densities, use regressors to infer it. In Kraskov's one kNN is used for it is statistical properties, however the learning capacity of kNN is limited given the size of the dataset while in theory MI should be evaluated with infinite capacity. Therefore these estimators, notably when there are strong dependencies and even more when such dependencies are non-linear, have a negative-bias and thus their estimates are lower-bounds to the through value.

These weaknesses become more striking as the dimensionality increases: using kNN, the estimators need an exponential number of samples and a tune of the input parameters. As such they suffer from the same curse of dimensionality that feature selection is supposed to solve.

A major overhaul of mutual information estimation seems necessary in order to apply our proposed algorithm efficiently.

Nonetheless there is plenty of possible improvements. For example instead of performing feature selection on the raw features it could be done on an encoding of them, as proposed in [14], after a feature extraction step. It could be of interest also the use of the feature extraction algorithm to somehow improve the mutual information estimator.

On the theoretical side a follow up could be the investigation of the possibility of upper bounding the Q-function error of the optimal policy with the Q-function error of a specific policy such as the most exploratory policy. If possible the bound would become of the highest relevance as it could limit the error on the optimal policy without trying to compute it.

On the other hand, a practical solution could be to embed the feature selection algorithm inside a RL model-free algorithm similarly to [17]. At each learning step, as for example in policy iteration, the greedy procedure could be run. In this way the dimensionality could be reduced during the learning process and at its end the bound would have been evaluated for the optimal policy that it has learned. Performance drawbacks could be relieved if it were possible to combine this technique with the above scheme for mutual information estimation.

Appendix A

Mutual Information Estimator

Here we introduce Kraskov's mutual information estimator [34], the local non-uniform correction proposed by [15] and then its improvement suggested in [3]. Finally we propose an adaptation of these correction terms for the conditional MI instance.

A.1 KSG estimator

The KSG estimator uses a generalization of the concept of sample-spacing from the one dimensional case to the d dimensional. Sample-spacing revolves around the idea of estimating a probability distribution density from the distances between nearest samples.

In 1987, [38] proposed an entropy estimator based on sample-spacing to directly estimate the expected log-density of a probability distribution through:

$$\mathbb{E}_{\mathbf{X}}\left[-\log p(\mathbf{x})\right] \approx \hat{\mathrm{H}}(\mathbf{X}) = \psi(N) - \psi(k) + \log(c_d) - \frac{d}{N}\sum_{i=1}^{N}\log \epsilon_{i,k} \quad \text{(A.1.1)}$$

where N is the sample size, ψ is the digamma function, $\epsilon_{i,k}$ are the distances from the i sample to its k^{th} nearest neighbour given a predefined norm for the space (usually max-norm or the Euclidian one) and c_d is the volume of the unit ball ($c_d = 1$ in max-norm). In d dimesions this estimator can be efficiently implemented through a well-known kNN algorithm such as KDTrees or Ball-trees and it could be pretty accurate as the only assumption made is the local uniformity of the density near the samples.

While the above estimator can be used in order to estimate the mutual information naively as sum of entropies as in eq. (2.4.7), the KSG estimator tries to reduce the bias that would arise from summing the entropies estimated with the same k, which would mean the use of different scales for joint and marginal spaces (a source of bias when the density is not uniform).

In order to avoid such bias, restricting us to max-norm, we can compute the joint entropy with a fixed k while we compute the marginal ones counting the points that are half the max distance of the k nearest neighobours away in each dimension $(n_j(i))$ and take it as the new k, leading to the estimator:

$$\hat{\mathrm{H}}(\mathbf{X}) = \psi(N) - \frac{1}{N}\sum_{i=1}^{N}\psi(n_j(i)+1) + \log(c_d) - \frac{d}{N}\sum_{i=1}^{N}\log\epsilon_{i,k} \tag{A.1.2}$$

Thus, KSG elides the mean distances term and this, generalized for the mutlivariate case, leads to:

$$\hat{\mathrm{I}}(\mathbf{X}) = (d-1)\psi(N) + \psi(k) - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{d}\psi(n_j(i)+1) - \frac{d-1}{k} \tag{A.1.3}$$

where the last term is just a correction one for consistency.

A.2 Limitation

This estimator however suffers from the curse of dimensionality: more and more samples are needed as d increases. This is due to the fact it relies on the idea of volumes in the joint space and that, in order to cover it evenly, a number of samples exponential in d must be collected.

Moreover it has another extremely relevant limitation: as proven in [15] the magnitude of the estimation is upper bounded by logarithm of the number of samples. This means that exponentially more samples are needed for strongly correlated features w.r.t. to weakly dependent ones.

A.3 LNC correction

The last limitation may be attributable to the assumption of local uniformity of denisty, and thus [15] propose a correction term to overcome it. The main idea is that for each data point there is a region, of volume $\bar{V}^i$, inside the max-rectangle, of volume V^i for which the local uniformity assumption is correct. Therefore, we can add a correction term taking into account this discrepancy:

$$\hat{\mathrm{I}}_{LNC}(\mathbf{X}) = \hat{\mathrm{I}}(\mathbf{X}) - \frac{1}{N}\sum_{i=1}^{N}\log\frac{\bar{V}^i}{V^i} \tag{A.3.1}$$

The last term increases as the uniform region, and its volume $\bar{V}^i$, becomes smaller, an event more probable when variables are correlated. Indeed, this increase is not bounded by N, mitigating the above issue.

In [15] the authors suggest the use of principal component analysis (PCA) for the estimation of the restricted volumes $\bar{V}^i$. The soundeness of such approach relies on the much weaker assumption of local linearity over the restricted region. On the other hand, the PCA volume may be smaller than the original one by chance as well. Therefore a trade-off parameter α is expected as input to the estimator to dictate for which datapoints adding the correction is reasonable. The input α is supposed to be the expected ratio of PCA and original volumes in the uniform distribution case, and thus the correction term is added when the computed ratio is smaller.

This correction is, however, inable to deal with redundant information and, consequently, also with higher dimensionalities because the PCA volume would decrease in such cases even when there is no correlation. A practical solution, proposed by [3], is to compute the original volume as the product of the PCA of the marginal ones:

$$\hat{\mathrm{I}}_{LNC^2}(\mathbf{X}) = \hat{\mathrm{I}}(\mathbf{X}) - \frac{1}{N}\sum_{i=1}^{N} \log \frac{\bar{V}^i}{\prod_{j=0}^{d} \bar{V}_j^i} \tag{A.3.2}$$

Even so the LNC estimator is not robust and has experimentally shown that it could overstimate the mutual information. It is a trade-off between accuracy on weakly dependent variables, where KSG shines, and in the strongly dependent case, where LNC overcomes KSG limitations.

A.4 Conditional MI adaptation

For the conditional mutual information quantity the LNC correction has not been previously defined so we propose its generalization to such problem.

Remark (LNC for CMI). We suggest a local non-uniformity correction, in the condition mutual information case, such that;

$$\hat{\mathrm{I}}_{LNC^2}(X;Y|Z) = \hat{\mathrm{I}}(X;Y|Z) - \frac{1}{N}\sum_{i=1}^{N} \log \frac{\bar{V}_{xy}^i \bar{V}_z^i}{\bar{V}_{xyz}^i} \tag{A.4.1}$$

Proof. In the multivariate case it is well known that:

$$\mathrm{I}\,(X;Y|Z) = \mathrm{I}\,(X;Y) - \mathrm{I}(X;Y;Z), \tag{A.4.2}$$

and the above corrections (eqs. (A.3.1) and (A.3.2)) work in the multivariate case as well. Thus, it is possible to combine the correction terms for the two mutual information without estimating both of them and with many simplifications as follows:

$$\begin{aligned}
\hat{\mathrm{I}}_{LNC^2}(X;Y|Z) &= \hat{\mathrm{I}}_{LNC^2}(X;Y) - \hat{\mathrm{I}}_{LNC^2}(X;Y;Z) \\
&= \hat{\mathrm{I}}(X;Y) - \hat{\mathrm{I}}(X;Y;Z) + \\
&\qquad - \frac{1}{N}\left(\sum_{i=1}^{N} \log \frac{\bar{V}_{xy}^i}{\bar{V}_x^i \bar{V}_y^i} - \sum_{i=1}^{N} \log \frac{\bar{V}_{xyz}^i}{\bar{V}_x^i \bar{V}_y^i \bar{V}_z^i}\right) \\
&\approx \hat{\mathrm{I}}(X;Y|Z) - \frac{1}{N}\sum_{i=1}^{N} \log \frac{\bar{V}_{xy}^i \bar{V}_z^i}{\bar{V}_{xyz}^i}
\end{aligned}$$

QED

This adaptation has the major advantage of allowing us to use directly the KSG estimator for CMI instead of the one for MI with a difference, which is optimized for such application.

Appendix B

Tables of the Tests

Here we present some of the tables from the tests on the three environments with different hyper-parameters and with an estimation of the true Q-function error for the purpose of bound verification.

Set of Selected Features	$\left\Vert Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\Vert_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$	0	0
$\{f_0, f_1, f_3, f_4, f_5, f_6, f_7\}$	5.0223e-07	0.037976
$\{f_0, f_1, f_3, f_4, f_5, f_7\}$	3.9091e-07	5.4884
$\{f_0, f_1, f_4, f_5, f_7\}$	3.8924e-07	13.548
$\{f_0, f_1, f_4, f_5\}$	4.5416e-07	46.459
$\{f_0, f_1, f_5\}$	6.036e-07	121.71
$\{f_0, f_1\}$	2.1876	179
$\{f_0\}$	7.8781	242.55
$\emptyset$	39.937	282.19

Table B.1: Verification of upper bound. Backward FS on LQG environment using bound in eq. (4.4.5), tested with $\gamma = 0.5, n = 500, k = 20$.

Set of Selected Features	$\left\Vert Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\Vert_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$	0	0
$\{f_0, f_1, f_3, f_4, f_5, f_6, f_7\}$	5.8318e-07	82.4
$\{f_0, f_1, f_3, f_4, f_5, f_7\}$	6.1474e-07	175.49
$\{f_0, f_1, f_3, f_4, f_5\}$	6.5371e-07	252.11
$\{f_0, f_1, f_4, f_5\}$	6.13e-07	290.14
$\{f_0, f_1, f_4\}$	0.28169	509.94
$\{f_0, f_4\}$	2.2457	633.72
$\{f_0\}$	58.419	764.25
$\emptyset$	97.289	842.84

Table B.2: Verification of upper bound. Backward FS on LQG environment using bound in eq. (4.4.5), tested with $\gamma = 0.9, n = 1000, k = 40$.

Set of Selected Features	$\left\Vert Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\Vert_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$	0	0
$\{f_0, f_1, f_2, f_4, f_5, f_6, f_7\}$	5.7547e-07	64.978
$\{f_0, f_1, f_2, f_4, f_5, f_6\}$	7.4872e-07	187.92
$\{f_0, f_1, f_4, f_5, f_6\}$	6.1886e-07	336.21
$\{f_0, f_1, f_4, f_5\}$	5.0254e-07	426.66
$\{f_0, f_1, f_5\}$	0.0077318	785.55
$\{f_1, f_5\}$	4.2011	929.89
$\{f_1\}$	104.25	1057.1
$\emptyset$	162.24	1143.6

Table B.3: Verification of upper bound. Backward FS on LQG environment using bound in eq. (4.4.5), tested with $\gamma = 0.95, n = 5000, k = 40$.

Set of Selected Features	$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	0	0
$\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	9.5007e-07	0.45563
$\{f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	1.0957e-06	0.64437
$\{f_2, f_3, f_4, f_5, f_7, f_8\}$	1.2552e-06	0.78919
$\{f_2, f_3, f_4, f_5, f_8\}$	1.006e-06	0.91128
$\{f_8, f_2, f_3, f_5\}$	1.8196e-06	1.0188
$\{f_8, f_2, f_3\}$	2.1432e-06	1.1161
$\{f_2, f_3\}$	2.1947e-06	9.5572
$\{f_3\}$	0.32068	13.312
$\emptyset$	3.6179	15.373

Table B.4: Verification of upper bound. Backward FS on LunarLander environment using bound in eq. (4.4.5), tested with $\gamma = 0.5, n = 500, k = 20$.

Set of Selected Features	$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	0	0
$\{f_0, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	1.2825e-06	3.3344
$\{f_0, f_2, f_3, f_4, f_6, f_7, f_8\}$	1.9925e-06	4.8078
$\{f_2, f_3, f_4, f_6, f_7, f_8\}$	3.0277e-06	5.9624
$\{f_2, f_3, f_4, f_7, f_8\}$	3.1337e-06	6.9517
$\{f_8, f_2, f_3, f_4\}$	2.8222e-06	7.8355
$\{f_8, f_2, f_3\}$	4.2191e-06	8.8411
$\{f_2, f_3\}$	3.9441e-06	24.864
$\{f_3\}$	2.0809	49.624
$\emptyset$	7.9208	60.7

Table B.5: Verification of upper bound. Backward FS on LunarLander environment using bound in eq. (4.4.5), tested with $\gamma = 0.9, n = 1000, k = 40$.

Set of Selected Features	$\left\|Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	0	0
$\{f_0, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$	1.5294e-06	2.1802
$\{f_0, f_2, f_3, f_5, f_6, f_7, f_8\}$	1.7211e-06	3.4016
$\{f_2, f_3, f_5, f_6, f_7, f_8\}$	2.5789e-06	4.4619
$\{f_2, f_3, f_5, f_7, f_8\}$	2.862e-06	5.4298
$\{f_8, f_2, f_3, f_5\}$	2.713e-06	6.3388
$\{f_8, f_2, f_3\}$	3.0588e-06	11.569
$\{f_2, f_3\}$	3.8655e-06	39.356
$\{f_3\}$	5.0803	96.098
$\emptyset$	12.702	123.89

Table B.6: Verification of upper bound. Backward FS on LunarLander environment using bound in eq. (4.4.5), tested with $\gamma = 0.95, n = 5000, k = 40$.

Set of Selected Features	$\left\|Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4\}$	0	0
$\{f_0, f_2, f_3, f_4\}$	0.75487	36.516
$\{f_2, f_3, f_4\}$	0.82994	43.807
$\{f_3, f_4\}$	0.83284	44.507
$\{f_4\}$	0.83406	45.091
$\emptyset$	4.2453	48.836

Table B.7: Verification of upper bound. Backward FS on CustomTaxi environment using bound in eq. (4.4.5), tested with $\gamma = 0.5, n = 500, k = 20$.

Set of Selected Features	$\left\|Q^{\pi}(\boldsymbol{\phi}) - \hat{Q}^{\pi}(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4\}$	0	0
$\{f_0, f_2, f_3, f_4\}$	1.476	181.26
$\{f_2, f_3, f_4\}$	1.6506	202.62
$\{f_2, f_4\}$	1.6681	203.93
$\{f_4\}$	1.6766	204.76
$\emptyset$	4.4977	210.36

Table B.8: Verification of upper bound. Backward FS on CustomTaxi environment using bound in eq. (4.4.5), tested with $\gamma = 0.9, n = 1000, k = 40$.

Set of Selected Features	$\left\|Q^\pi(\boldsymbol{\phi}) - \hat{Q}^\pi(\boldsymbol{\phi}_S)\right\|_{2,\mu}$	Bound
$\{f_0, f_1, f_2, f_3, f_4\}$	0	0
$\{f_0, f_2, f_3, f_4\}$	2.031	188.55
$\{f_2, f_3, f_4\}$	2.2608	214.11
$\{f_2, f_4\}$	2.3047	222.11
$\{f_4\}$	2.325	223.62
$\emptyset$	5.9298	233.06

Table B.9: Verification of upper bound. Backward FS on CustomTaxi environment using bound in eq. (4.4.5), tested with $\gamma = 0.95, n = 5000, k = 40$.

Bibliography

[1] B. Behzadian, S. Gharatappeh, and M. Petrik, "Fast feature selection for linear value function approximation", 2019 (cit. on pp. 15, 36, 39, 43).

[2] M. Beraha, A. M. Metelli, M. Papini, A. Tirinzoni, and M. Restelli, "Feature selection via mutual information: New theoretical insights", *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2019 (cit. on pp. 2, 11, 18, 19, 21, 28, 29).

[3] N. Carrara and J. Ernst, "On the estimation of mutual information", *arXiv preprint arXiv:1910.00365*, 2019 (cit. on pp. 34, 45, 47).

[4] F. M. Garcia and P. S. Thomas, "A meta-mdp approach to exploration for lifelong reinforcement learning", in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019 (cit. on p. 36).

[5] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker, "On variational bounds of mutual information", in *International Conference on Machine Learning*, 2019 (cit. on p. 25).

[6] G. Ver Steeg, *Npeet*, Dec. 18, 2019. [Online]. Available: `https://github.com/gregversteeg/NPEET` (cit. on p. 34).

[7] K. Asadi, D. Misra, and M. L. Littman, "Lipschitz continuity in model-based reinforcement learning", in *ICML*, 2018 (cit. on p. 21).

[8] B. Behzadian and M. Petrik, "Low-rank feature selection for reinforcement learning.", in *ISAIM*, 2018 (cit. on p. 15).

[9] D. McAllester and K. Statos, "Formal limitations on the measurement of mutual information", *arXiv preprint arXiv:1811.04251*, 2018 (cit. on p. 34).

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition, ser. Adaptive computation and machine learning series. The MIT Press, 2018, ISBN: 9780262039246 (cit. on pp. 4, 5).

[11] S. Raschka and V. Mirjalili, *Python Machine Learning*, 2nd ed. Packt Publishing, 2017 (cit. on p. 12).

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: `arXiv:1606.01540` (cit. on p. 36).

[13] S. Shen and M. Chi, "Aim low: Correlation-based feature selection for model-based reinforcement learning.", *International Educational Data Mining Society*, 2016 (cit. on p. 16).

[14] Z. Song, R. E. Parr, X. Liao, and L. Carin, "Linear feature encoding for reinforcement learning", in *NIPS*, 2016 (cit. on pp. 12, 13, 15, 43).

[15] S. Gao, G. Ver Steeg, and A. Galstyan, "Efficient estimation of mutual information for strongly dependent variables", in *Artificial intelligence and statistics*, 2015 (cit. on pp. 34, 45, 46).

[16] D.-R. Liu, H.-L. Li, and D. Wang, "Feature selection and feature learning for high-dimensional batch reinforcement learning: A survey", *International Journal of Automation and Computing*, vol. 12, pp. 229–242, 2015 (cit. on p. 19).

[17] S. Loscalzo, R. Wright, and L. Yu, "Predictive feature selection for genetic policy search", *Autonomous Agents and Multi-Agent Systems*, 2015 (cit. on p. 44).

[18] R. W. Wright, X. Qiao, S. Loscalzo, and L. Yu, "Improving approximate value iteration with complex returns by bounding", in *AAAI*, 2015 (cit. on p. 9).

[19] A. Lucini Paioni, "Controllo di un gruppo di ascensori tramite apprendimento per rinforzo con selezione delle variabili", Master's thesis, Politecnico di Milano, 2014. [Online]. Available: `http://hdl.handle.net/10589/92450` (cit. on p. 2).

[20] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection", *Journal of machine learning research*, 2012 (cit. on pp. 19, 28).

[21] C. Painter-Wakefield and R. Parr, "Greedy algorithms for sparse reinforcement learning", in *ICML*, 2012 (cit. on pp. 2, 14).

[22] A. Castelletti, S. Galelli, M. Restelli, and R. Soncini-Sessa, "Tree-based variable selection for dimensionality reduction of large-scale control systems", *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 62–69, 2011 (cit. on pp. 2, 14, 16, 21).

[23] H. Hachiya and M. Sugiyama, "Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information", in *ECML/PKDD*, 2010 (cit. on pp. 1, 13, 16, 21, 41).

[24] J. Johns, C. Painter-Wakefield, and R. Parr, “Linear complementarity for regularized policy evaluation and improvement”, in *Advances in neural information processing systems*, 2010 (cit. on p. 13).

[25] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein, “Feature selection using regularization in approximate linear programs for markov decision processes”, *arXiv preprint arXiv:1005.1860*, 2010 (cit. on p. 13).

[26] J. Z. Kolter and A. Y. Ng, “Regularization and feature selection in least-squares temporal difference learning”, in *ICML*, 2009 (cit. on p. 13).

[27] M. Kroon and S. Whiteson, “Automatic feature selection for model-based reinforcement learning in factored mdps”, in *2009 International Conference on Machine Learning and Applications*, 2009 (cit. on p. 16).

[28] J. Morimoto, S.-H. Hyon, C. G. Atkeson, and G. Cheng, “Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction”, in *2008 IEEE International Conference on Robotics and Automation*, 2008 (cit. on p. 16).

[29] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman, “An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning”, in *ICML*, 2008 (cit. on pp. 13, 19).

[30] C. Sourav, *Lecture 2: Distances between probability measures*, UC Berkley, 2008 (cit. on p. 10).

[31] R. parr, C. Painter-Wakefield, L. Li, and M. L. Littman, “Analyzing feature generation for value-function approximation”, in *ICML*, 2007 (cit. on p. 14).

[32] L. Li, T. J. Walsh, and M. L. Littman, “Towards a unified theory of state abstraction for mdps.”, in *ISAIM*, 2006 (cit. on p. 1).

[33] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Second edition. John Wiley & Sons, Inc., 2005 (cit. on pp. 4, 9).

[34] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information”, *Physical review E*, 2004 (cit. on pp. 34, 45).

[35] A. L. Gibbs and F. E. Su, “On choosing and bounding probability metrics”, *International statistical review*, vol. 70, no. 3, pp. 419–435, 2002 (cit. on p. 10).

[36] J. Peters, “Policy gradient methods for control applications”, Technical Report TR-CLMC-2007-1, University of Southern California, Tech. Rep., 2002 (cit. on p. 36).

[37] L. Breiman, “Random forests”, *Machine learning*, 2001 (cit. on p. 39).

[38] L. Kozachenko and N. N. Leonenko, “Sample estimate of the entropy of a random vector”, *Problemy Peredachi Informatsii*, 1987 (cit. on p. 45).

[39] C. E. Shannon, “A mathematical theory of communication”, *Bell System Technical Journal*, 1948 (cit. on p. 9).