Mach Learn (2011) 82: 399–443
DOI 10.1007/s10994-010-5225-4

Knows what it knows: a framework for self-aware learning

Lihong Li · Michael L. Littman · Thomas J. Walsh · Alexander L. Strehl

Received: 7 August 2009 / Revised: 12 October 2010 / Accepted: 30 October 2010 /
Published online: 25 November 2010

Abstract We introduce a learning framework that combines elements of the well-known PAC and mistake-bound models. The KWIK (knows what it knows) framework was designed particularly for its utility in learning settings where active exploration can impact the training examples the learner is exposed to, as is true in reinforcement-learning and active-learning problems. We catalog several KWIK-learnable classes as well as open problems, and demonstrate their applications in experience-efficient reinforcement learning.

Keywords Reinforcement learning · Knows What It Knows (KWIK) · Probably Approximately Correct (PAC) · Mistake bound · Computational learning theory · Exploration · Active learning

1 Motivation

At the core of recent reinforcement-learning (RL) algorithms that enjoy polynomial sample complexity guarantees (Kearns and Singh 2002; Kearns and Koller 1999; Brafman and Ten-

Editor: Roni Khardon.

Part of the work was done while L. Li, T. Walsh, and A. Strehl were at the Rutgers University.

L. Li (✉)
Yahoo! Research, 4401 Great America Parkway, Santa Clara, CA 95054, USA
e-mail: lihong@yahoo-inc.com

M.L. Littman
Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854, USA
e-mail: mlittman@cs.rutgers.edu

T.J. Walsh
Department of Computer Science, University of Arizona, 1040 E. 4th Street, Tucson, AZ 85721, USA
e-mail: twalsh@cs.arizona.edu

A.L. Strehl
Facebook, 1601 S California Ave, Palo Alto, CA 94304, USA
e-mail: astrehl@facebook.com

nenholtz 2002; Kakade et al. 2003; Strehl et al. 2006a, 2007) lies the idea of distinguishing between instances that have been learned with sufficient accuracy and those whose outputs are still unknown.

The Rmax algorithm (Brafman and Tennenholtz 2002), for example, estimates transition probabilities for each state–action–next-state triple of a Markov decision process. The estimates are made separately, as licensed by the Markov property, and the accuracy of the estimate is bounded using Hoeffding bounds. The algorithm explicitly distinguishes between probabilities that have been estimated accurately (known) and those for which more experience will be needed (unknown). By encouraging the agent to gather more experience in the unknown states, Rmax can guarantee a polynomial bound on the number of timesteps in which it has a non-near-optimal policy (Kakade 2003).

In this paper, we make explicit the properties that are sufficient for a learning algorithm to be used in efficient exploration algorithms like Rmax. Roughly, the learning algorithm needs to make only accurate predictions, although it can opt out of predictions by saying "I don't know" ($\perp$). However, there must be a (polynomial) bound on the number of times the algorithm can respond $\perp$. We call such a learning algorithm KWIK ("know what it knows").

Section 2 provides a motivating example and sketches possible uses for KWIK algorithms. Section 3 defines the KWIK conditions more precisely and relates them to established models from computational learning theory. Sections 4–6 survey a set of hypothesis classes for which KWIK algorithms can be created. Section 7 describes a number of applications of KWIK in reinforcement learning, focusing on unifying existing results as well as providing a new algorithm for factored-state Markov decision processes. Section 8 concludes the paper and lists a few important open problems.

The research described in this paper builds upon and extends the work presented at the Twenty-Fifth International Conference on Machine Learning (Li et al. 2008) and in Advances in Neural Information Processing Systems 20 (Strehl and Littman 2008).

2 A KWIK example

Consider the simple navigation task in Fig. 1. There is a set of nodes connected by edges, with the node on the left as the source and the dark one on the right as the sink. Each edge in the graph is associated with a binary *cost vector* of dimension $n = 3$, indicated in the figure. The cost of traversing an edge is the dot product of its cost vector with a fixed weight vector $\theta = [1, 2, 0]$. Assume that θ is not known to the agent, but the graph topology and all cost vectors are. In each episode, the agent starts from the source and moves along some path to the sink. Each time it crosses an edge, the agent observes its true cost. The cost of a path from the source to the sink is the sum of edge costs along the path. Once the sink is reached, the next episode begins. Through navigating in the graph, the agent tries to identify an optimal path with minimum cost while taking a suboptimal path in as few episodes as

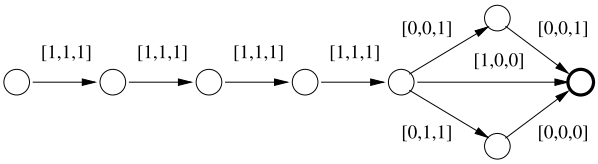

Fig. 1 A cost-vector navigation graph

possible. There are 3 distinct paths in this example. Given the θ value above, the top has a cost of 12, the middle 13, and the bottom 15.

A simple approach for this task is for the agent to assume edge costs are uniform and walk the shortest (middle) path to collect data. It would gather 4 examples of $[1, 1, 1] \rightarrow 3$ and one of $[1, 0, 0] \rightarrow 1$. Standard regression algorithms could use this dataset to find a $\hat{\theta}$ that fits this data perfectly. However, there may be several $\hat{\theta}$s supported by the available data. Here, the least-squares solution, $\hat{\theta} = [1, 1, 1]$, is a natural choice. The learned weight vector could then be used to estimate costs for the three paths: 14 for the top, 13 for the middle, 14 for the bottom. Using these estimates, an agent would continue to take the middle path forever, never realizing it is not optimal.

In contrast, consider a learning algorithm that "knows what it knows". Instead of creating an approximate weight vector $\hat{\theta}$, it reasons about whether the costs for each edge can be obtained from the available data. The middle path, since all its edge costs have been observed, is definitely 13. The last edge of the bottom path has cost vector $[0, 0, 0]$, so its cost must be zero, but the penultimate edge of this path has cost vector $[0, 1, 1]$. This vector is a linear combination of the two observed cost vectors, so, regardless of θ, its cost is

$$\theta \cdot [0, 1, 1] = \theta \cdot ([1, 1, 1] - [1, 0, 0]) = \theta \cdot [1, 1, 1] - \theta \cdot [1, 0, 0],$$

which is just $3 - 1 = 2$. Thus, the agent knows the bottom path's cost is 14—worse than the middle path.

The vector $[0, 0, 1]$ on the top path is linearly independent of the observed cost vectors, so its cost is undecided. We know we don't know. A safe thing to assume provisionally is that the cost is zero—the smallest possible cost, encouraging the agent to try the top path in the second episode. Now, it observes $[0, 0, 1] \rightarrow 0$, allowing it to solve for θ exactly ($\theta = [1, 2, 0]$) and accurately predict the cost for any vector (since the training data spans $\Re^n$). It now knows that it knows all the costs, and can confidently take the optimal (top) path.

In general, any algorithm that guesses a weight vector may never find the optimal path. An algorithm that uses linear algebra to distinguish known from unknown costs will either take an optimal route *or* discover the cost of a linearly independent cost vector on each episode. Thus, it can never choose suboptimal paths more than n times. Formal discussions of learning noise-free linear functions are provided in Sect. 4.2.

In contrast, an agent that does not generalize, but visits every edge to learn its cost, will require m episodes to learn optimal behavior, in the worst case, where m is the number of edges in the graph. This example shows how combining generalization with explicitly distinguished known and unknown areas can lead to efficient and optimal decision algorithms.

The motivation for studying KWIK learning grew out of its use in sequential decision making problems like this one. However, other machine-learning problems could benefit from this perspective and from the development of efficient algorithms. For instance, action selection in bandit problems (Fong 1995a) and associative bandit problems (Strehl et al. 2006c) (bandit problems with inputs) can both be addressed in the KWIK setting by choosing the better arm when both payoffs are known and an unknown arm otherwise. KWIK could also be a useful framework for studying active learning (Cohn et al. 1994) and anomaly detection (Lane and Brodley 2003), both of which are machine-learning problems that require some degree of reasoning about whether a recently presented input is predictable from previous examples. When mistakes are costly, as in utility-based data mining (Weiss and Tian 2006) or learning robust control (Bagnell et al. 2001), having explicit predictions of certainty can be very useful for decision making.

3 Formal definition and related frameworks

This section provides a formal definition of KWIK learning and its relationship to existing frameworks. Particular attention is paid to the popular Probably Approximately Correct (PAC) and Mistake Bound (MB) frameworks.

3.1 KWIK definition

KWIK is an objective for supervised-learning algorithms. In particular, we begin with an *input set* $\mathcal{X}$ and *output set* $\mathcal{Y}$. The *hypothesis class* $\mathcal{H}$ consists of a set of functions from $\mathcal{X}$ to $\mathcal{Y}$: $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y})$. The *target function* $h^* : \mathcal{X} \to \mathcal{Y}$ is the source of training examples and is unknown to the learner.

Two parties are involved in a KWIK learning protocol. The *learner* runs a learning algorithm and makes predictions; while the *environment*, which represents an instance of a KWIK learning problem, provides the learner with inputs and observations. The protocol for a KWIK "run" is as follows:

- The hypothesis class $\mathcal{H}$, accuracy parameter ϵ, and confidence parameter δ are known to both the learner and the environment.
- The environment selects a target function h^* *adversarially*.
- For *timestep* $t = 1, 2, 3, \ldots$,
  - The environment selects an input $x_t \in \mathcal{X}$ *adversarially* and informs the learner. The target value $y_t = h^*(x_t)$ is unknown to the learner.
  - The learner predicts an output $\hat{y}_t \in \mathcal{Y} \cup \{\bot\}$. We call $\hat{y}_t$ *valid* if $\hat{y} \neq \bot$.
  - If $\hat{y}_t = \bot$, the learner makes an observation $z_t \in \mathcal{Z}$ of the output. In the deterministic case, $z_t = y_t$, but generally the relation between z_t and y_t is problem specific. For example, $z_t = 1$ with probability y_t and 0 with probability $1 - y_t$ in the Bernoulli case, and $z_t = y_t + \eta_t$ for a zero-mean random variable η_t in the additive noise case.

Definition 1 Let $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y})$ be a hypothesis class. We say that $\mathcal{H}$ is *KWIK-learnable* if there exists an algorithm $\mathbf{A}$ with the following property: for any $0 < \epsilon, \delta < 1$, the following two requirements are satisfied with probability at least $1 - \delta$ in a whole run of $\mathbf{A}$ according to the KWIK protocol above:

1. (*Accuracy Requirement*) If $h^* \in \mathcal{H}$, then all non-$\bot$ predictions must be ϵ-accurate; that is, $|\hat{y}_t - y_t| < \epsilon$ whenever $\hat{y}_t \neq \bot$;[1]
2. (*Sample Complexity Requirement*) The total number of $\bot$s predicted during the whole run, denoted $B(\epsilon, \delta, \dim(\mathcal{H}))$, is bounded by a function polynomial in $1/\epsilon$, $1/\delta$, and $\dim(\mathcal{H})$, where $\dim(\mathcal{H})$ is a pre-defined nonnegative-valued function measuring the *dimension* or *complexity* of $\mathcal{H}$.

We call $\mathbf{A}$ a *KWIK algorithm* and $B(\epsilon, \delta, \dim(\mathcal{H}))$ a *KWIK bound* of $\mathbf{A}$. We will also use $B(\epsilon, \delta)$ for simplicity, if there is no ambiguity. Furthermore, $\mathcal{H}$ is *efficiently KWIK-learnable* if the per-timestep time complexity of $\mathbf{A}$ is polynomial in $1/\epsilon$, $1/\delta$, and $\dim(\mathcal{H})$.

[1] Here, $|\hat{y} - y|$ may be any nonnegative-valued function that measures the *discrepancy* or *distance* between $\hat{y}$ and y. To simplify notation, we have used the operator "$-$" in a broad sense and it is not restricted to algebraic subtraction. For instance: (i) in the special situation of $\mathcal{Y} \subseteq \Re$, as is considered in many cases of this paper, $|\cdot|$ is understood as the absolute value; (ii) when $\mathcal{Y} \subseteq \Re^k$, $|\cdot|$ may mean vector norm; and (iii) when y and $\hat{y}$ are probability distributions, $|\cdot|$ may be used for total variation.

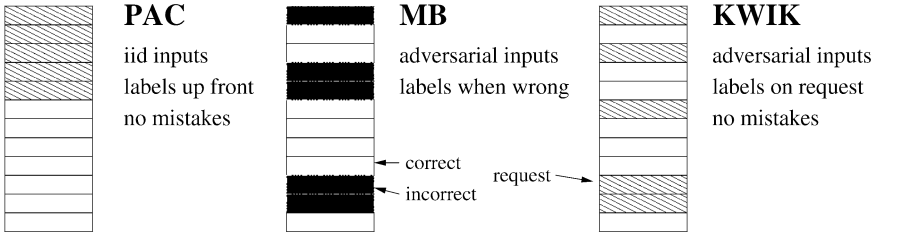

Fig. 2 Relationship of KWIK to a variant of the PAC and the MB frameworks in terms of how labels are provided for inputs

A few notes are in order. First, the accuracy requirement in Definition 1 considers the "realizable" case only; namely, the target function is assumed to be in the hypothesis class. In contrast, the sample-complexity requirement has to hold even in the "unrealizable" case where $h^* \notin \mathcal{H}$.

Second, all KWIK algorithms provided in this paper enjoy polynomial time complexity and thus are efficient. We note that a KWIK algorithm can be randomized, and so the confidence parameter, δ, may be necessary even in deterministic problems.

Third, the definition above allows the hypothesis class $\mathcal{H}$ to be infinite. In this case, we must require the KWIK bound to be polynomial in the complexity measure, $\dim(\mathcal{H})$, rather than in the size of $\mathcal{H}$. While in some situations (such as in Sect. 5) $\dim(\mathcal{H})$ is straightforward to define, it remains an open question what measure would be appropriate for KWIK in general. More discussions are in Sect. 5.

3.2 Connection to PAC and MB

Figure 2 illustrates the relationship of KWIK to the similar *PAC* (Probably Approximately Correct) (Valiant 1984) and *MB* (Mistake Bound) (Littlestone 1987) frameworks. In all three cases, a series of inputs (instances) is presented to the learner. Each input is depicted in the figure by a rectangular box.

In the PAC model, inputs are drawn from a fixed distribution, and the learner is provided with labels (correct outputs) for an initial sequence of *i.i.d.* inputs, depicted by shaded rectangles. After that point, the learner is required to produce a hypothesis that, with probability at least $1-\delta$, makes accurate predictions for $1-\epsilon$ fraction of inputs. For the purpose of the present paper, it is natural to consider a variant of PAC, where the learner is required to produce an ϵ-accurate outputs (empty boxes) for all new inputs with probability at least $1-\delta$.

In the MB model, the learner is expected to produce an output for every input. Labels are provided to the learner whenever it makes a mistake (filled boxes). Inputs are selected adversarially, so there is no bound on when the last mistake might be made. However, MB algorithms guarantee that the *total* number of mistakes is small, so the ratio of incorrect to correct outputs must go to zero asymptotically. In the most widely studied *concept learning* problem, any MB algorithm for a hypothesis class can be used to provide a PAC algorithm for the same class (Littlestone 1989),[2] but the reverse is not necessarily true (Blum 1994).

The KWIK model has elements of both PAC and MB. Like PAC, a KWIK algorithm has tolerance parameters allowing for some failures. Like MB, inputs to a KWIK algorithm

[2] An extension from concept learning to real-valued function learning is given by Klasner and Simon (1995).

may be selected adversarially. Instead of bounding mistakes, a KWIK algorithm must have a bound on the number of label requests ($\perp$) it can make. By requiring performance to be independent of how inputs are selected, a KWIK algorithm can be used in cases in which the input distribution is dependent in complex ways on the KWIK algorithm's behavior, as can happen in on-line or active learning settings.

Any KWIK algorithm can be turned into a MB algorithm with the same bound by simply having the algorithm guess an output each time it is not certain; the KWIK bound is automatically an upper bound of the number of mistakes. However, some hypothesis classes are exponentially harder to learn in the KWIK setting than in the MB setting. An example is conjunctions of n Boolean variables, in which MB algorithms can guess "false" when uncertain and learn with $n+1$ mistakes, but a KWIK algorithm may need $\Omega(2^n)$ many $\perp$s to acquire the negative examples required to capture the target hypothesis: a concrete sequence of adversarial input is given in Li (2009), and a related lower bound is obtained by Angluin (1988) for the "double sunflower" lemma in the query-based learning model.

The separation is not only theoretically important, but illustrates an important differences in the allowable behavior and cost of information for MB and KWIK. In both MB and KWIK, we can consider the learning process as a two-player game where at even depth nodes of the game tree the adversary picks a new input and a label for the last input that must be consistent with the nodes above it. At the odd nodes, the learner gets to make a prediction. The game terminates when the odd player has formed the correct hypothesis and the payout is determined as follows. For MB, the payout is the negative of the number of mistakes predictions along the trajectory. For KWIK, the payout is $-B(\epsilon, \delta, \dim(\mathcal{H}))$, based on the number of $\perp$s on the chosen trajectory in the game tree, plus $-\infty$ for any trajectory that contains a mistaken (with probability at least δ) prediction.

Notice that *structurally* the MB tree and the KWIK tree are virtually identical (except for the $\perp$ action), but that their payout structures will induce very different behavior. For instance, in the case of learning a conjunction, if the adversary chooses a positive example, future trajectories (assuming the adversary plays rationally) in both the MB and KWIK tree will be exponentially smaller than under the corresponding negative example, which provides very little information. However, an optimal agent in the MB tree can mitigate cost on these long (negative) trajectories by always selecting "false" when it does not have evidence that the chosen input is true. This will either cost it a single mistake but put it on a very short branch (getting a highly informative positive example), or cost it nothing but put it on a very long (but not costly) path. On the other hand, the optimal agent in the KWIK game tree does not have the option of hedging its bets. Where mistakes were made in the MB tree with only a local unit cost, in the KWIK tree the payout structure makes them infinitely expensive. The only guaranteed finite payout available to the agent comes from choosing $\perp$ on every input with uncertain output, and there are exponentially many of these as the adversary can force the agent down one of the longer (negative example only) paths.

Intuitively, the MB agent can declare its intentions after the adversary announces the label. If it correctly predicted a negative example, MB "knew it all the time". If it was wrong, it made a highly informative mistake. In contrast, the KWIK agent cannot engage in such self deception. To avoid the $-\infty$ payout, it must only make valid predictions when it is absolutely certain of the outcome. The benefit of this restriction, is that unlike the MB agent, the KWIK agent is intrinsically "self aware": before it makes a prediction, it can always categorize the step as "exploration" ($\perp$) or "exploitation" (label), while the MB algorithm inherently relies on making such distinctions after the fact. While such devotion to certainty may seem too restrictive, we argue in Sect. 7.3 that the lack of assumptions and self awareness in KWIK is a better fit than either MB or PAC for creating sample-efficient reinforcement learning algorithms.

3.3 Other online-learning models

The notion of allowing the learner to opt out of some inputs by returning $\perp$ is not unique to KWIK. Several other authors have considered related models:

- *Sleeping Experts* (Freund et al. 1997a) can respond $\perp$ for some inputs, although they need not learn from these experiences and the number of $\perp$s in the whole run may be unbounded;
- Learners in the settings of *Selective Sampling* (Cesa-Bianchi et al. 2006) and *Label Efficient Prediction* (Cesa-Bianchi et al. 2005) request labels randomly with a changing probability and achieve bounds on the expected number of mistakes and the expected number of label requests for a finite number of interactions. These algorithms cannot be used unmodified in the KWIK setting because, with high probability, KWIK algorithms must not make mistakes at any time;
- In the MB-like *Apple-Tasting* setting (Helmbold et al. 2000), the learner receives feedback asymmetrically only when it predicts a particular label (a positive example, say), which conflates the request for a sample with the prediction of a particular outcome.
- *Query-by-committee* algorithms (Seung et al. 1992; Freund et al. 1997b) determine the confidence level of their predictions for an example based on the degree of disagreements in the predictions of sub-algorithms in the committee. Similarly to selective sampling algorithms, query-by-committee algorithms were proposed for active learning and may not possess the same accuracy requirement as KWIK.
- Finally, the Averaged Classifier by Freund et al. (2004) may return $\perp$ if the averaged prediction for an example is close to the classification boundary. However, it is assumed that examples are *i.i.d.* and a non-$\perp$ prediction may not be accurate.

Conformal Prediction (Shafer and Vovk 2008) is an online learning paradigm in which the learner has to predict a *region*, rather than a *point* prediction, for the present input based on previously observed input–output pairs. It is required that these prediction regions contain the correct output with high probability. It is straightforward to decide whether the output is known within sufficient accuracy based on the "size" of the region. For example, in regression problems, if the region is an interval of length smaller than ϵ, then any point prediction in this region will be ϵ-accurate with high probability. However, existing conformal prediction methods make statistical assumptions about inputs such as independence or exchangeability, and thus are rendered inapplicable in the KWIK setting.

Another framework similar to KWIK is the *regret* framework as applied to the *Associative Bandit Problem* (Strehl et al. 2006c): at every timestep t in this model, the learner receives input $x_t \in \mathcal{X}$, selects an action a_t from a possibly infinite set $\mathcal{A}$, and then receives a randomized payoff $r_t \in \Re$, whose expectation depends on x_t and a_t. The goal is to minimize the regret, defined as the difference between the largest total payoff by following the best action-selection rule π in some given rule set Π and the total payoff received by the learner. *No-regret algorithms* have been found for variants of associative bandit problems in the sense that their regrets are sublinear in the number of timesteps (*e.g.*, Auer 2002). Consequently, the average per-timestep regret converges to 0 in the limit. However, these algorithms do not satisfy the KWIK requirements for exactly the same reason that MB algorithms are not KWIK: the learner does not know for certain when the last mistake is made, even if the overall probability of mistaken predictions is tiny.

4 Some KWIK-learnable classes

This section describes some hypothesis classes for which KWIK algorithms are available. It is not meant to be an exhaustive survey, but simply to provide a flavor for the properties of hypothesis classes KWIK algorithms can exploit. The complexity of many learning problems has been characterized by defining the *dimensionality* of hypothesis classes (Angluin 2004). No such definition has been found for the KWIK model, so we resort to enumerating examples of learnable classes.

4.1 Memorization and enumeration

We begin by describing the simplest and most general KWIK algorithms.

Problem 1 *The* `memorization` *algorithm can learn any hypothesis class with input set* $\mathcal{X}$ *with a KWIK bound of* $|\mathcal{X}|$. *This algorithm can be used when the input set* $\mathcal{X}$ *is finite and observations are noise free.*

To achieve this bound, the algorithm simply keeps a mapping $\hat{h}$ initialized to $\hat{h}(x) = \perp$ for all $x \in \mathcal{X}$. When the environment chooses an input x, the algorithm reports $\hat{h}(x)$. If $\hat{h}(x) = \perp$, the environment will provide a label y and the algorithm will assign $\hat{h}(x) := y$. It will only report $\perp$ once for each input, so the KWIK bound is $|\mathcal{X}|$.

Problem 2 *The* `enumeration` *algorithm can learn any hypothesis class* $\mathcal{H}$ *with a KWIK bound of* $|\mathcal{H}| - 1$. *This algorithm can be used when the hypothesis class* $\mathcal{H}$ *is finite and observations are noise free.*

The algorithm keeps track of $\hat{\mathcal{H}}$, the version space, and initially $\hat{\mathcal{H}} = \mathcal{H}$. Each time the environment provides input $x \in \mathcal{X}$, the algorithm computes $\hat{L} = \{h(x) \mid h \in \hat{\mathcal{H}}\}$. That is, it builds the set of all outputs for x for all hypotheses that have not yet been ruled out. If $|\hat{L}| = 0$, the version space has been exhausted and the target hypothesis is not in the hypothesis class ($h^* \notin \mathcal{H}$).

If $|\hat{L}| = 1$, it means that all hypotheses left in $\hat{\mathcal{H}}$ agree on the output for this input, and therefore the algorithm knows what the proper output must be. It returns $\hat{y} \in \hat{L}$. On the other hand, if $|\hat{L}| > 1$, two hypotheses in the version space disagree. In this case, the algorithm returns $\perp$ and receives the true label y. It then computes an updated version space

$$\hat{\mathcal{H}}' = \{h \mid h \in \hat{\mathcal{H}} \wedge h(x) = y\}.$$

Because $|\hat{L}| > 1$, there must be some $h \in \hat{\mathcal{H}}$ such that $h(x) \neq y$. Therefore, the new version space must be smaller: $|\hat{\mathcal{H}}'| \leq |\hat{\mathcal{H}}| - 1$. Before the next input is received, the version space is updated $\hat{\mathcal{H}} := \hat{\mathcal{H}}'$.

If $|\hat{\mathcal{H}}| = 1$ at any point, $|\hat{L}| = 1$, and the algorithm will no longer return $\perp$. Therefore, $|\mathcal{H}| - 1$ is the maximum number of $\perp$s the algorithm can return.

Example 1 You own a bar that is frequented by a group of n patrons P. There is one patron $\mathbf{f} \in P$ who is an instigator—whenever a group of patrons is in the bar $G \subseteq P$, if $\mathbf{f} \in G$, a fight will break out. However, there is another patron $\mathbf{p} \in P$, who is a peacemaker. If $\mathbf{p}$ is in the group, it will prevent a fight, even if $\mathbf{f}$ is present.

You want to predict whether a fight will break out among a subset of patrons, initially without knowing the identities of **f** and **p**. The input set is $\mathcal{X} = 2^P$ and the output set is $\mathcal{Y} = \{\text{fight, no fight}\}$.

The memorization algorithm achieves a KWIK bound of 2^n for this problem, since it may have to see each possible subset of patrons. However, the enumeration algorithm can KWIK-learn this hypothesis class with a bound of $n(n-1)-1$ since there is one hypothesis for each possible assignment of a patron to **f** and **p**. Each time it reports $\perp$, it is able to rule out at least one possible instigator–peacemaker combination.

4.2 Real-valued functions

The previous two algorithms exploited the finiteness of the hypothesis class and input set. KWIK bounds can also be achieved when these sets are infinite.

Problem 3 *Define* $\mathcal{X} = \Re^n$, $\mathcal{Y} = \Re$, *and*

$$\mathcal{H} = \{f \mid f(x) = \theta \cdot x, \theta \in \Re^n\}.$$

That is, $\mathcal{H}$ *is a set of linear functions on* n *variables for some unknown weight vector* θ. *In the deterministic case where* $z_t = y_t$, $\mathcal{H}$ *can be KWIK-learned by the algorithm* `deterministic linear regression` *with a KWIK bound of* $B(\epsilon, \delta) = n$.

The algorithm maintains a training set $\mathcal{T}$ of training examples, which is initialized to the empty set $\emptyset$ prior to learning. On the tth input x_t, let $\mathcal{T} = \{(v_1, f(v_1)), (v_2, f(v_2)), \ldots, (v_k, f(v_k))\}$ be the current set of training examples. The algorithm first detects if x_t is linearly independent of the previous inputs stored in $\mathcal{T}$. This can be done efficiently by, say, Gaussian elimination (Golub and Van Loan 1989). If x_t is linearly independent, then the algorithm predicts $\perp$, observes the output $y_t = f(x_t)$, and then expands the training set: $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x_t, y_t)\}$. Otherwise, there exist k real numbers, $a_1, a_2, \ldots, a_k$, such that $x_t = a_1 v_1 + a_2 v_2 + \cdots + a_k v_k$. In this case, we can accurately predict the value of $f(x_t)$ using linear algebra:

$$\begin{aligned} f(x_t) &= \theta \cdot x_t \\ &= \theta \cdot (a_1 v_1 + a_2 v_2 + \cdots + a_k v_k) \\ &= a_1 \theta \cdot v_1 + a_2 \theta \cdot v_2 + \cdots + a_k \theta \cdot v_k \\ &= a_1 f(v_1) + a_2 f(v_2) + \cdots + a_k f(v_k). \end{aligned}$$

By operation of the algorithm, the inputs in $\mathcal{T}$ (that is, $v_1, v_2, \ldots, v_k$) must be linearly independent at all times. Hence, $\mathcal{T}$ contains at most n training pairs, so the algorithm will predict $\perp$ at most n times. Any case where the inputs contain n such linearly independent vectors shows this KWIK bound is tight.

The `deterministic linear regression` algorithm above may be used by the `distance learning` below to learn the ℓ_2-distance between two points.

Problem 4 *Define* $\mathcal{X} = \Re^n$, $\mathcal{Y} = \Re$, *and*

$$\mathcal{H} = \{f \mid f(x) = \|x - c\|, c \in \Re^n\},$$

where $\|x\| = \sqrt{x^\mathsf{T} x}$ *denotes the* ℓ_2 *norm of vector* $x \in \Re^n$. *That is, there is an unknown point and the target function maps input points to the Euclidean distance from the unknown point. The* `distance learning` *algorithm can learn in this hypothesis class with a KWIK bound of* $n + 1$.

Although this problem can be solved using a geometric argument (Li et al. 2008), it is easier to describe the solution via a reduction to `deterministic linear regression`.[3] We start with the squared ℓ_2-distance of an input x to the unknown point c: $\|x - c\|^2 = \|c\|^2 - 2c^\mathsf{T} x + \|x\|^2$, and rewrite it into $\|x - c\|^2 - \|x\|^2 = \|c\|^2 - 2c^\mathsf{T} x$. The right-hand side may be viewed as a linear function, $\bar{c}^\mathsf{T} \bar{x}$, where $\bar{x}$ and $\bar{c}$ are the augmented input and weight vectors, respectively:

$$\bar{x} := \begin{bmatrix} x \\ 1 \end{bmatrix}, \qquad \bar{c} := \begin{bmatrix} -2c \\ \|c\|^2 \end{bmatrix}.$$

Therefore, we may use `deterministic linear regression` to KWIK-learn the function, $\|x - c\|^2 - \|x\|^2$, with a KWIK bound of $n + 1$. If we can make an accurate prediction for this function, we can easily compute the distance $\|x - c\|$.

4.3 Noisy observations

Up to this point, observations have been noise free. In this subsection, we consider a couple of noisy KWIK learning problems. We start with the simplest Bernoulli case and note that the same algorithm can actually be applied to KWIK-learn the expectation of a bounded, real-valued random variable.

Problem 5 *The* `coin learning` *algorithm can accurately predict the probability that a biased coin will come up heads given Bernoulli observations with a KWIK bound of*

$$B(\epsilon, \delta) = \frac{1}{2\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right).$$

We have a biased coin whose unknown probability of heads is p. In the notation of this paper, $\mathcal{X}$ is a singleton containing an arbitrarily chosen element (meaning that we have a single coin with a fixed but unknown head probability), $\mathcal{Y} = [0, 1]$, and $\mathcal{Z} = \{0, 1\}$ with 0 for tail and 1 for head. We want to learn an estimate $\hat{p}$ that is accurate ($|\hat{p} - p| \leq \epsilon$) with high probability ($1 - \delta$).

If we could observe p, then this problem would be trivial: Say $\perp$ once, observe p, and let $\hat{p} = p$. The KWIK bound is thus 1. Now, however, observations are noisy. Instead of observing p, we see either 1 (with probability p) or 0 (with probability $1 - p$).

Each time the algorithm says $\perp$, it gets an independent trial that it can use to compute the empirical probability: $\hat{p} = \frac{1}{T} \sum_{t=1}^{T} z_t$, where $z_t \in \mathcal{Z}$ is the tth observation in T trials. The number of trials needed before we are $1 - \delta$ certain our estimate is within ϵ-accuracy can be computed using a Hoeffding (1963) bound:

$$T = \frac{1}{2\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right).$$

[3] This approach was suggested by Robert Schapire.

The coin-learning problem above can be regarded as a special case of the dice-learning problem. Here, the agent is to KWIK-learn a multinomial distribution over n elements, where each distribution is specified by n non-negative numbers that sum up to unity. The prediction error is defined as the total variance between the true distribution and the learner's prediction; given two discrete distributions, y and $\hat{y}$, over the same discrete sample space, Ω, their total variation is defined as half of their ℓ_1 distance: $|y - \hat{y}| := \frac{1}{2} \sum_{\omega \in \Omega} |y(\omega) - \hat{y}(\omega)|$.

Problem 6 *The* `dice learning` *algorithm can accurately predict a multinomial distribution over n elements within ϵ total variation given multinomial observations with a KWIK bound of*

$$B(\epsilon, \delta) = \frac{n}{8\epsilon^2} \ln \frac{2n}{\delta} = O\left(\frac{n}{\epsilon^2} \ln \frac{n}{\delta}\right).$$

The algorithm is almost identical to `coin learning`. It first draws T samples from the multinomial distribution and then uses the empirical distribution for prediction.

To derive a KWIK bound for `dice learning`, we may use `coin learning` to KWIK-learn each probability in the multinomial distribution $h = (h(\omega_1), h(\omega_2), \ldots, h(\omega_n))$ to ensure ϵ/n accuracy with probability at least $1 - \delta/n$. Then, an application of the union bound yields the following KWIK bound for `dice learning`

$$O\left(\frac{n^2}{\epsilon^2} \ln \frac{n}{\delta}\right),$$

which is asymptotically worse than the stated bound. However, a more careful analysis using the multiplicative form of Chernoff's bound can prove the stated KWIK bound. A proof is given by Kakade (2003).

The `dice learning` algorithm will serve as an important building block in many of our applications to reinforcement learning. The important case of noisy linear functions is studied in Sect. 5.

5 Learning noisy linear functions

This section extends the `deterministic linear regression` algorithm (Problem 3) to the noisy case where the observations are target outputs corrupted by additive, white noise. The algorithm we present here is based on previous work by Strehl and Littman (2008), who first formulated and solved this problem.

Recently, two related algorithms were developed. Walsh et al. (2009) considered a slightly different setting than the one defined in Sect. 3. Specifically, their algorithm can always see the label even if it makes a valid prediction, although a later refinement of this work (available as a technical report) performed the analysis under the same conditions as the algorithm above. The sample complexity results for this algorithm were on par with those reported here, but because it employed regularization, the computation and storage requirements were both polynomial in the number of dimensions, rather than growing with the number of samples as an naive implementation of the algorithm above would. In that light, this later work can be viewed as a practical version of the one presented here, but with slightly different mechanics.

Cesa-Bianchi et al. (2009) studied selective sampling for binary classification with linear classifiers. Under a weaker assumption that the environment is an *oblivious* adversary

(namely, the sequence of inputs is fixed beforehand), their `Parametric BBQ` algorithm achieves a significantly better KWIK bound of $\tilde{O}(d/\epsilon^2)$, where $\tilde{O}(\cdot)$ suppresses logarithmic factors. The KWIK bound we present in this section, however, remains valid even when the environment is *adaptive*, that is, when the environment has the ability to decide the next input based on previous interaction history with the learner. Furthermore, their bound has a logarithmic dependence on the total number of timesteps. In contrast, our KWIK bound here (as well as the one by Walsh et al. 2009) does not have this dependence—a property that is necessary for the KWIK algorithm's application to reinforcement learning in Sect. 7.2.

5.1 Problem formulation

In the noisy linear regression problem, certain regularity assumptions are necessary to make it feasible. Recall that for a vector $x \in \Re^n$, we denote its ℓ_2 norm by $\|x\|$.

Problem 7 *Define* $\mathcal{X} = \{x \in \Re^n \mid \|x\| \leq 1\}$, $\mathcal{Y} = \mathcal{Z} = [-1, 1]$, *and*

$$\mathcal{H} = \{f \mid f(x) = \theta^\mathsf{T} x, \theta \in \Re^n, \|\theta\| \leq 1\}.$$

That is, $\mathcal{H}$ *is the set of linear functions in* n *variables with bounded weight vectors. The target output is corrupted by additive, bounded, white noise:* $z_t = y_t + \eta_t$, *where* η_t *is a bounded random variable with zero expectation. Note that no further assumption is made on the distribution of* η_t. *The* `noisy linear regression` *algorithm* (*Algorithm* 1) *can learn in* $\mathcal{H}$ *with a KWIK bound of* $B(\epsilon, \delta) = \tilde{O}(n^3/\epsilon^4)$.

The deterministic case was described in Sect. 4.2 with a KWIK bound of n. Here, the algorithm must be cautious to average over the noisy samples to make predictions accurately. The algorithm uses the least-squares estimate of the weight vector for inputs with high certainty. Certainty is measured by two terms representing (1) the number and proximity of previous samples to the current input and (2) the appropriateness of the previous samples for making a least-squares estimate. When certainty is low for either measure, the algorithm reports $\perp$ and observes a noisy sample of the linear function.

5.2 Solution

Let $X \in \Re^{m \times n}$ denote an $m \times n$ matrix whose rows we interpret as transposed input vectors. We let $X(i)$ denote the transpose of the ith row of X. Let $z \in \Re^m$ denote an m-dimensional vector whose ith component, denoted $z(i)$, is interpreted as the corresponding noisy observation.

Since $X^\mathsf{T}X$ is symmetric and positive semi-definite, it can be written as the following form of *singular value decomposition* (Golub and Van Loan 1989):

$$X^\mathsf{T}X = U \Lambda U^\mathsf{T}, \tag{1}$$

where $U = [u_1, \ldots, u_n] \in \Re^{n \times n}$, with $u_1, \ldots, u_n$ being a set of orthonormal singular vectors of $X^\mathsf{T}X$, and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, with corresponding singular values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 1 > \lambda_{k+1} \geq \cdots \geq \lambda_n \geq 0$. Note that Λ is diagonal but not necessarily invertible. Now, define $\bar{U} = [u_1, \ldots, u_k] \in \Re^{n \times k}$ and $\bar{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_k) \in \Re^{k \times k}$. For a fixed input x_t (a new input provided to the algorithm at time t), define

$$\bar{q} := X \bar{U} \bar{\Lambda}^{-1} \bar{U}^\mathsf{T} x_t \in \Re^m, \tag{2}$$

$$\bar{u} := \left[0, \ldots, 0, u_{k+1}^\mathsf{T} x_t, \ldots, u_n^\mathsf{T} x_t\right]^\mathsf{T} \in \Re^n. \tag{3}$$

Our algorithm uses these quantities and is provided in pseudocode by Algorithm 1. The main result about this algorithm is the following theorem.

Theorem 1 *With appropriate parameter settings, Algorithm 1 is an efficient KWIK algorithm with a KWIK bound of* $\tilde{O}(n^3/\epsilon^4)$.

Although the analysis of Algorithm 1 is somewhat complicated, the algorithm itself has a natural interpretation. Given a new input x_t, the algorithm considers making a prediction of the output y_t using the norm-constrained least-squares estimator (specifically, $\hat{\theta}$ defined in line 6 of Algorithm 1). The norms of the vectors $\bar{q}$ and $\bar{u}$ provide a quantitative measure of uncertainty about this estimate. When both norms are small, the estimate is trusted and a valid prediction is made. When either norm is large, the estimate is not trusted and the algorithm produces an output of $\perp$.

The quantities $\bar{q}$ and $\bar{u}$ provide a measure of uncertainty for the least-squares estimate (Auer 2002). Consider the case when all eigenvalues of $X^\mathsf{T}X$ are greater than 1. In this case, note that $x = X^\mathsf{T}X(X^\mathsf{T}X)^{-1}x = X^\mathsf{T}\bar{q}$. Thus, x can be written as a linear combination, whose coefficients make up $\bar{q}$, of the rows of X, which are previously experienced inputs. Intuitively, if the norm of $\bar{q}$ is small, then there are many previous training samples (actually, combinations of inputs) "similar" to x, hence their noises cancel each other, and our least-squares estimate is likely to be accurate for x. For the case of ill-conditioned $X^\mathsf{T}X$ (when $X^\mathsf{T}X$ has singular values close to 0), $X(X^\mathsf{T}X)^{-1}x$ may be undefined or have a large norm. In this case, we must consider the directions corresponding to small singular values separately via $\bar{u}$: if $\|\bar{u}\|$ is sufficiently small, this direction can be ignored without significantly affecting the prediction; otherwise, the algorithm predicts $\perp$.

5.3 Analysis

Our analysis of Algorithm 1 hinges on two key lemmas that we now present. They together show that the cumulative squared error of a predictor can be used to bound its prediction error on a new input, a critical insight used to prove Theorem 1. The complete proofs are given in Appendix A.

Algorithm 1 `Noisy Linear Regression`

0: **Inputs:** α_1, α_2
1: Initialize $X = [\,]$ and $z = [\,]$.
2: **for** $t = 1, 2, 3, \ldots$ **do**
3: Let x_t denote the input at timestep t.
4: Compute $\bar{q}$ and $\bar{u}$ using Equations 2 and 3.
5: **if** $\|\bar{q}\| \leq \alpha_1$ and $\|\bar{u}\| \leq \alpha_2$ **then**
6: Choose $\hat{\theta} \in \Re^n$ that minimizes $\sum_i (z(i) - \bar{\theta}^\mathsf{T}X(i))^2$ subject to $\|\bar{\theta}\| \leq 1$.
7: Predict $\hat{y}_t = \hat{\theta}^\mathsf{T}x_t$.
8: **else**
9: Predict $\hat{y}_t = \perp$.
10: Receive observation z_t.
11: Append x_t^T as a new row to the matrix X.
12: Append z_t as a new element to the vector z.
13: **end if**
14: **end for**

The first lemma analyzes the behavior of the squared error of predictions based on an incorrect estimate $\hat{\theta}$ ($\neq \theta$) versus the squared error obtained by using the true parameter vector θ. Specifically, we show that the squared error of the former is very likely to be larger than the latter when the predictions based on $\hat{\theta}$ (of the form $\hat{\theta}^\mathsf{T} x$ for input x) are highly inaccurate. The proof makes use of Hoeffding (1963)'s bound.

Lemma 1 *Let $\theta \in \Re^n$ and $\hat{\theta} \in \Re^n$ be two fixed parameter vectors satisfying $\|\theta\| \leq 1$ and $\|\hat{\theta}\| \leq 1$. Suppose that $(x_1, z_1), \ldots, (x_m, z_m)$ is any sequence of samples satisfying $x_i \in \Re^n$, $\|x_i\| \leq 1$, $z_i \in [-1, 1]$, $\mathbf{E}[z_i \mid x_i] = \theta^\mathsf{T} x_i$, and $\mathbf{Var}[z_i \mid x_i] = \sigma_i^2$. For any $0 < \delta < 1$ and fixed positive constant w, if*

$$\sum_{i=1}^{m} \left((\theta - \hat{\theta})^\mathsf{T} x_i\right)^2 \geq 2\sqrt{8m \ln(2/\delta)} + w,$$

then

$$\sum_{i=1}^{m} \left(z_i - \hat{\theta}^\mathsf{T} x_i\right)^2 > \sum_{i=1}^{m} \left(z_i - \theta^\mathsf{T} x_i\right)^2 + w$$

with probability at least $1 - 2\delta$.

The next lemma relates the error of an estimate $\hat{\theta}^\mathsf{T} x$ for a fixed input x based on an incorrect estimate $\hat{\theta}$ to the quantities $\|\bar{q}\|$, $\|\bar{u}\|$, and $\Delta_{\mathrm{E}}(\hat{\theta}) := \sqrt{\sum_{i=1}^{m} ((\theta - \hat{\theta})^\mathsf{T} X(i))^2}$. Recall that when $\|\bar{q}\|$ and $\|\bar{u}\|$ are both small, our algorithm becomes confident of the least-squares estimate. In precisely this case, the lemma shows that $|(\theta - \hat{\theta})^\mathsf{T} x|$ is bounded by a quantity proportional to $\Delta_{\mathrm{E}}(\hat{\theta})$. This result justifies the condition used by Algorithm 1 to predict $\perp$.

Lemma 2 *Let $\theta \in \Re^n$ and $\hat{\theta} \in \Re^n$ be two fixed parameter vectors satisfying $\|\theta\| \leq 1$ and $\|\hat{\theta}\| \leq 1$. Suppose that $(x_1, z_1), \ldots, (x_m, z_m)$ is any sequence of samples satisfying $x_i \in \Re^n$, $\|x_i\| \leq 1$, and $z_i \in [-1, 1]$. Let $x \in \Re^n$ be any vector. Let $\bar{q}$ and $\bar{u}$ be defined as above. Let $\Delta_{\mathrm{E}}(\hat{\theta})$ denote the error term $\sqrt{\sum_{i=1}^{m} ((\theta - \hat{\theta})^\mathsf{T} x_i)^2}$. We have that*

$$|(\theta - \hat{\theta})^\mathsf{T} x| \leq \|\bar{q}\| \Delta_{\mathrm{E}}(\hat{\theta}) + 2\|\bar{u}\|.$$

6 Combining KWIK learners

This section provides examples of how KWIK learners can be combined to provide learning guarantees for more complex hypothesis classes. Their applications in reinforcement learning are the topic of the next section. We first consider a variant of Problem 1 that combines learners across disjoint input sets.

Problem 8 *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be a set of disjoint input sets ($\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ if $i \neq j$) and $\mathcal{Y}$ be an output set. Let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be a set of KWIK-learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $\mathcal{H}_i \subseteq (\mathcal{X}_i \to \mathcal{Y})$. The* `input partition` *algorithm can learn the hypothesis class $\mathcal{H} \subseteq (\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_k \to \mathcal{Y})$ with a KWIK bound of $B(\epsilon, \delta) = \sum_i B_i(\epsilon, \delta/k)$.*

The `input partition` algorithm runs the learning algorithm, denoted $\mathbf{A}_i$, for each subclass $\mathcal{H}_i$ using parameters ϵ and δ/k. When it receives an input $x \in \mathcal{X}_i$, it queries $\mathbf{A}_i$ and returns its response $\hat{y}$. If $\hat{y} = \perp$, an observation is obtained and `input partition` informs $\mathbf{A}_i$ of $\hat{y}$ to allow it to learn. Otherwise, $\hat{y}$ is ϵ-accurate with high probability, as guaranteed by individual subalgorithms. The total number of $\perp$s is the sum of the number of $\perp$s returned by all subalgorithms $\mathbf{A}_i$. To achieve $1-\delta$ certainty, it insists on $1-\delta/k$ certainty from each of the subalgorithms. By a union bound, the overall failure probability must be less than the sum of the failure probabilities for the subalgorithms, which is at most δ.

Example 2 Let $\mathcal{X} = \mathcal{Y} = \Re$. Define $\mathcal{H}$ to be a set of piecewise linear functions:

$$\mathcal{H} = \{f \mid f(x) = px \text{ if } x \geq 0,\ f(x) = mx \text{ otherwise}, p \in \Re, m \in \Re\}.$$

Using `deterministic linear regression` (Problem 3), we can KWIK-learn the class of linear functions over two input sets, $\mathcal{X}_- = (-\infty, 0)$ and $\mathcal{X}_+ = [0, \infty)$, each requiring a KWIK bound of 1. Note that $\{\mathcal{X}_-, \mathcal{X}_+\}$ is a *partition* of the entire input set: $\mathcal{X}_- \cup \mathcal{X}_+ = \mathcal{X}$ and $\mathcal{X}_- \cap \mathcal{X}_+ = \emptyset$. We can use `input partition` to KWIK-learn $\mathcal{H}$ with a KWIK bound of $1+1=2$. The two KWIK learners are called $\mathbf{A}_-$ and $\mathbf{A}_+$, respectively.

Assume the first input is $x_1 = 2$, which is in $\mathcal{X}_+$. The `input partition` algorithm queries $\mathbf{A}_+$ with input x_1. Since $\mathbf{A}_+$ has no idea about y_1, it returns $\perp$. Hence, `input partition` reports $\hat{y}_1 = \perp$, and $y_1 = 4$ is observed. Learner $\mathbf{A}_+$ can now infer with certainty that $p = y_1/x_1 = 2$, and we can now predict $f(x)$ for all $x \in \mathcal{X}_+$. The next input is $x_2 = 0.5 \in \mathcal{X}_+$, which is again presented to $\mathbf{A}_+$, resulting in $\hat{y}_2 = px_2 = 1$. The third input is $x_3 = -1 \in \mathcal{X}_-$. The algorithm queries $\mathbf{A}_-$ with input x_3 and receives $\perp$ since $\mathbf{A}_-$ does not have enough information to predict y_3. The algorithm then predicts $\perp$ for the second time and sees $y_3 = 3$. Learner $\mathbf{A}_-$ can now determine $m = y_3/x_3 = -3$, and the target function is completely identified:

$$f(x) = \begin{cases} 2x & \text{if } x \geq 0, \\ -3x & \text{otherwise.} \end{cases}$$

A similar approach applies to the output set $\mathcal{Y}$ when it is a cross product of k sets: $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$. Since the accuracy requirement in Definition 1 depends on the output set as well as the interpretation of the discrepancy metric $|\cdot|$, it is necessary to make certain assumptions to relate the discrepancy metric of $\mathcal{Y}$ to those of $\mathcal{Y}_i$. A natural choice, which we will use in Sect. 7, is to assume the existence of some $\alpha \in (0, 1]$ such that for any $\epsilon \in (0, 1)$ and any $y_i, \hat{y}_i \in \mathcal{Y}_i$, if $|\hat{y}_i - y_i| < \alpha\epsilon$ for all $i = 1, 2, \ldots, k$, then $|\hat{y} - y| < \epsilon$, where $y = (y_1, \ldots, y_k)$ and $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_k)$. For example, if we use the ℓ_1, ℓ_2, or ℓ_∞ norms in the output spaces $\mathcal{Y}$ and $\mathcal{Y}_i$, then α can be $1/k$, $1/\sqrt{k}$, and 1, respectively.

Problem 9 *Let $\mathcal{X}$ be an input set and $\mathcal{Y}_1, \ldots, \mathcal{Y}_k$ be a collection of output sets. Let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be a set of KWIK-learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $\mathcal{H}_i \subseteq (\mathcal{X} \to \mathcal{Y}_i)$. The* `output combination` *algorithm can KWIK-learn the hypothesis class $\mathcal{H} \subseteq (\mathcal{X} \to \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k)$ with a KWIK bound of $B(\epsilon, \delta) = \sum_i B_i(\alpha\epsilon, \delta/k)$.*

The next algorithm generalizes the previous algorithm by combining both the input and output sets.

Problem 10 *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ and $\mathcal{Y}_1, \ldots, \mathcal{Y}_k$ be a set of input and output sets and $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be a collection of KWIK-learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ on these sets. That is, $\mathcal{H}_i \subseteq (\mathcal{X}_i \to \mathcal{Y}_i)$. The* `cross product` *algorithm can learn the hypothesis class $\mathcal{H} \subseteq ((\mathcal{X}_1 \times \cdots \times \mathcal{X}_k) \to (\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k))$ with a KWIK bound of $B(\epsilon, \delta) = \sum_i B_i(\alpha\epsilon, \delta/k)$.*

Here, each input consists of a vector of inputs from each of the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ and outputs are vectors of outputs from $\mathcal{Y}_1, \ldots, \mathcal{Y}_k$. Like Problem 9, each component of this vector can be learned independently via the corresponding algorithm. Each is learned to within an accuracy of $\alpha\epsilon$ and confidence $1 - \delta/k$. Any time any component returns $\perp$, the `cross product` algorithm returns $\perp$. Since each $\perp$ returned can be traced to one of the subalgorithms, the total is bounded as described above. By the union bound, total failure probability is no more than $k \times \delta/k = \delta$. We note that Problem 9 is a degenerate case in which $\mathcal{X}_i = \mathcal{X}_j$ for all i, j and each input vector contains the same element in its components.

Example 3 You own a chain of k bars, each of which is frequented by a set of n patrons, P_i, and once again each bar has an instigator $\mathbf{f}_i \in P_i$ and a peacemaker $\mathbf{p}_i \in P_i$. The patron sets do not overlap. On any given night, with some subset of each P_i at each bar, you need to predict whether, for each bar, there will be a fight. Notice that `input partition` cannot solve this problem because a prediction must be made for each bar, and `output combination` is similarly insufficient because there are k disjoint input sets. But using `cross product` with `enumeration` (Problem 2) for each bar achieves a KWIK bound of $k(n(n-1)-1)$. We note that generally `cross product` can be wrapped around several different learners (e.g. `enumeration` for one sub-problem and `memorization` for another).

The previous two algorithms concern combinations of input or output sets and apply to both deterministic and noisy observations. We next provide an intuitive algorithm for the deterministic case that combines hypothesis classes.

Problem 11 *Let $\mathcal{F} \subseteq (\mathcal{X} \to \mathcal{Y})$ be the set of functions mapping input set $\mathcal{X}$ to output set $\mathcal{Y}$. Let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be a set of KWIK-learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $\mathcal{H}_i \subseteq \mathcal{F}$ for all $1 \leq i \leq k$. That is, all the hypothesis classes share the same input/output sets. The* `union` *algorithm can learn the joint hypothesis class $\mathcal{H} = \bigcup_i \mathcal{H}_i$ with a KWIK bound of $B(\epsilon, \delta) = (k-1) + \sum_i B_i(\epsilon/2, \delta/k)$.*

The `union` algorithm is like a higher-level version of `enumeration` (Problem 2) and applies in the deterministic setting. It maintains a set of active algorithms $\hat{A}$, one for each hypothesis class: $\hat{A} = \{1, \ldots, k\}$. Given an input x, the `union` algorithm queries each algorithm $i \in \hat{A}$ to obtain a prediction $\hat{y}_i$ from each active algorithm. Let $\hat{L} = \{\hat{y}_i \mid i \in \hat{A}\}$. Furthermore, a union bound implies that, with probability at least $1 - \delta$, all predictions of $\mathbf{A}_i$ must be $\epsilon/2$-accurate whenever $h^* \in \mathcal{H}_i$.

If $\perp \in \hat{L}$, the `union` algorithm reports $\perp$ and obtains the correct output y. Any algorithm i for which $\hat{y} = \perp$ is then sent the correct output y to allow it to learn. When $\perp \notin \hat{L}$, we consider two cases as follows.

If $\hat{L}$ is consistent, meaning that there exists $\tilde{y} \in \mathcal{Y}$ such that $\max_{\bar{y} \in \hat{L}} |\bar{y} - \tilde{y}| < \epsilon/2$, then we claim $\tilde{y}$ is ϵ-accurate with high probability: by assumption, one of the predictions in $\hat{L}$,

denoted $\bar{y}^*$, must be $\epsilon/2$-accurate, and thus

$$|\tilde{y} - y| \leq |\tilde{y} - \bar{y}^*| + |\bar{y}^* - y| \leq \epsilon/2 + \epsilon/2 = \epsilon.$$

If, on the other hand, $\hat{L}$ is not consistent, meaning that there is no $\tilde{y} \in \mathcal{Y}$ such that $\max_{\bar{y} \in \hat{L}} |\bar{y} - \tilde{y}| < \epsilon/2$, then `union` returns $\perp$ and obtains the correct output y. Any algorithm that makes a prediction error greater than $\epsilon/2$ is eliminated for future consideration. That is,

$$\hat{A}' = \{i \mid i \in \hat{A} \wedge (\hat{y}_i = \perp \vee |\hat{y}_i - y| > \epsilon/2)\}.$$

Clearly, at least one algorithm will be eliminated because any $y \in \mathcal{Y}$ must differ from at least one element in $\hat{L}$ by at least $\epsilon/2$.

On each input for which the `union` algorithm reports $\perp$, either one of the subalgorithms reported $\perp$ (at most $\sum_i B_i(\epsilon/2, \delta/k)$ times) or two algorithms disagreed and at least one was removed from $\hat{A}$ (at most $k-1$ times). The KWIK bound follows from these facts.

Example 4 Let $\mathcal{X} = \mathcal{Y} = \Re$. Now, define $\mathcal{H}_1 = \{f \mid f(x) = |x - c|, c \in \Re\}$. That is, each function in $\mathcal{H}_1$ maps x to its distance from some unknown point c. We can learn $\mathcal{H}_1$ with a KWIK bound of 2 using a 1-dimensional version of `distance learning`. Next, define $\mathcal{H}_2 = \{f \mid f(x) = mx + b, m \in \Re, b \in \Re\}$. That is, $\mathcal{H}_2$ is the set of lines. We can learn $\mathcal{H}_2$ with a KWIK bound of 2 using `deterministic linear regression`. Finally, define $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2$, the union of these two classes. We can use `union` to KWIK-learn $\mathcal{H}$.

Assume the first input is $x_1 = 2$. The `union` algorithm queries the learners for $\mathcal{H}_1$ and $\mathcal{H}_2$ with x_1 as input and neither has any idea, so it returns $\perp$ and receives the feedback $y_1 = 2$, which it passes to the subalgorithms. The next input is $x_2 = 8$. The learners for $\mathcal{H}_1$ and $\mathcal{H}_2$ still don't have enough information, so it returns $\perp$ and sees $y_2 = 4$, which it passes to the subalgorithms. Next, $x_3 = 1$. Now, the learner for $\mathcal{H}_1$ unambiguously computes $c = 4$, because that's the only interpretation consistent with the first two examples ($|2 - 4| = 2$, $|8 - 4| = 4$), so it returns $|1 - 4| = 3$. On the other hand, the learner for $\mathcal{H}_2$ unambiguously computes $m = 1/3$ and $b = 4/3$, because that's the only interpretation consistent with the first two examples ($2 \times 1/3 + 4/3 = 2$, $8 \times 1/3 + 4/3 = 4$), so it returns $1 \times 1/3 + 4/3 = 5/3$. Since the two subalgorithms disagree, the `union` algorithm returns $\perp$ one last time and finds out that $y_3 = 3$. It makes all future predictions (accurately) using the algorithm for $\mathcal{H}_1$.

Finally, we provide a powerful algorithm that generalizes the `union` algorithm to work with noisy observations as well. The basic idea is similar to the method for hypothesis testing of Kearns and Schapire (1994), who consider a PAC-style learning model for probabilistic concepts.

Problem 12 *Let $\mathcal{F} \subseteq (\mathcal{X} \to \mathcal{Y})$ be the set of functions mapping input set $\mathcal{X}$ to output set $\mathcal{Y} = [0, 1]$. Let $\mathcal{Z} = \{0, 1\}$ be a binary observation set. Let $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be a set of KWIK-learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $\mathcal{H}_i \subseteq \mathcal{F}$ for all $1 \leq i \leq k$. That is, all the hypothesis classes share the same input/output sets. The* `noisy union` *algorithm can learn the joint hypothesis class $\mathcal{H} = \bigcup_i \mathcal{H}_i$ with a KWIK bound of*

$$B(\epsilon, \delta) = O\left(\frac{k}{\epsilon^2} \ln \frac{k}{\delta}\right) + \sum_{i=1}^{k} B_i\left(\frac{\epsilon}{8}, \frac{\delta}{k+1}\right).$$

We first sketch the special case of $k = 2$ to explain the intuition of `noisy union`, and then extend the analysis to the general case. This algorithm is similar to the `union` algorithm (Problem 11), except that it has to deal with noisy observations. The algorithm proceeds by running the KWIK algorithms, denoted $\mathbf{A}_1$ and $\mathbf{A}_2$, using parameters (ϵ_0, δ_0), as subalgorithms for each of the $\mathcal{H}_i$ hypothesis classes, where $\epsilon_0 = \epsilon/4$ and $\delta_0 = \delta/3$. Given an input x_t in trial t, it queries each algorithm i to obtain a prediction $\hat{y}_{ti}$. Let $\hat{L}_t$ be the set of responses.

If $\perp \in \hat{L}_t$, the `noisy union` algorithm reports $\perp$, obtains an observation $z_t \in \mathcal{Z}$, and sends it to subalgorithms i with $\hat{y}_{ti} = \perp$ to allow them to learn. In the following, we focus on the remaining cases where $\perp \notin \hat{L}_t$.

If $|\hat{y}_{t1} - \hat{y}_{t2}| \leq 4\epsilon_0$, then these two predictions are sufficiently consistent, with high probability the prediction $\hat{p}_t = (\hat{y}_{t1} + \hat{y}_{t2})/2$ is ϵ-close to $y_t = \Pr(z_t = 1)$. This is the case because one of the predictions, say $\hat{y}_{t1}$, deviates from y_t by at most ϵ_0 with probability at least $1 - \delta/3$, and hence $|\hat{p}_t - y_t| = |\hat{p}_t - \hat{y}_{t1} + \hat{y}_{t1} - y_t| \leq |\hat{p}_t - \hat{y}_{t1}| + |\hat{y}_{t1} - \hat{y}_t| = |\hat{y}_{t1} - \hat{y}_{t2}|/2 + |\hat{y}_{t1} - \hat{y}_t| \leq 2\epsilon_0 + \epsilon_0 < \epsilon$.

If $|\hat{y}_{t1} - \hat{y}_{t2}| > 4\epsilon_0$, then the individual predictions are not sufficiently consistent for `noisy union` to make an ϵ-accurate prediction. Thus, it reports $\perp$ and needs to know which subalgorithm provided an inaccurate response. But, since the observations are noisy in this problem, it cannot eliminate h_i on the basis of a single observation. Instead, it maintains the total squared prediction error for every subalgorithm i: $e_i = \sum_{t \in \mathcal{T}} (\hat{y}_{ti} - z_t)^2$, where $\mathcal{T} = \{t \mid |\hat{y}_{t1} - \hat{y}_{t2}| > 4\epsilon_0\}$ is the set of trials in which the subalgorithms gave inconsistent predictions. Our last step is to show that e_i is a robust measure for eliminating invalid predictors when $|\mathcal{T}|$ is sufficiently large.

Applying the Hoeffding bound and some algebra, we find

$$\Pr(e_1 > e_2) \leq \exp\left(-\frac{\sum_{t \in \mathcal{T}} |\hat{y}_{t1} - \hat{y}_{t2}|^2}{8}\right) \leq \exp\left(-2\epsilon_0^2 |\mathcal{T}|\right).$$

Setting the righthand side to be $\delta/3$ and solving for $|\mathcal{T}|$, we have

$$|\mathcal{T}| = \frac{1}{2\epsilon_0^2} \ln \frac{3}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right).$$

Since each $\mathbf{A}_i$ succeeds with probability $1 - \delta/3$, and the comparison of e_1 and e_2 also succeeds with probability $1 - \delta/3$, a union bound implies that the `noisy union` algorithm succeeds with probability at least $1 - \delta$. All $\perp$s are either from a subalgorithm (at most $\sum_i B_i(\epsilon_0, \delta_0)$) or from the `noisy union` algorithm ($O(1/\epsilon^2 \ln(1/\delta))$).

The general case where $k > 2$ can be reduced to the $k = 2$ case by pairing the k learners and running the `noisy union` algorithm described above on each pair. Here, each subalgorithm is run with parameters $\epsilon/8$ and $\delta/(k+1)$. The algorithm is formally described in Algorithm 2. Although there are $\binom{k}{2} = O(k^2)$ pairs, a more careful analysis can reduce the dependence of the KWIK bound on k from quadratic to linearithmic,[4] leading to Theorem 2, whose proof is given in Appendix B. Note we have not attempted to optimize the constant factor in parameter m, but the resulting KWIK bound is the best possible up to a constant (Diuk et al. 2009).

[4] A function $f(n)$ is called *linearithmic* if $f(n) = O(n \ln n)$.

Algorithm 2 `Noisy Union`

Inputs: $\mathbf{A}_1, \ldots, \mathbf{A}_k, \epsilon, \delta$, m.
Run each subalgorithm $\mathbf{A}_i$ with parameters $\epsilon/8$ and $\delta/(k+1)$.
$R \leftarrow \{1, 2, \ldots, k\}$
for all $1 \leq i < j \leq k$ **do**
  $c_{ij} \leftarrow 0$
  $\Delta_{ij} \leftarrow 0$
end for
for timestep $t = 1, 2, \ldots$ **do**
  Observe $x_t \in \mathcal{X}$
  Run each $\mathbf{A}_i$ to obtain their predictions, $\hat{y}_{ti}$
  if $\hat{y}_{ti} = \perp$ for some $i \in R$ **then**
    Predict $\hat{y}_t = \perp$ and observe $z_t \in \mathcal{Z}$
    Send z_t to all subalgorithms $\mathbf{A}_i$ with $\hat{y}_{ti} = \perp$
  else
    if $|\hat{y}_{ti} - \hat{y}_{tj}| < \epsilon$ for all $i, j \in R$ **then**
      Predict the midpoint of the set of predictions: $\hat{y}_t = (\max_{i \in R} \hat{y}_{ti} + \min_{i \in R} \hat{y}_{ti})/2$
    else
      Predict $\hat{y}_t = \perp$ and observe $z_t \in \mathcal{Z}$
      for all $i, j \in R$ such that $i < j$ and $|\hat{y}_{ti} - \hat{y}_{tj}| \geq \epsilon/2$ **do**
        $c_{ij} \leftarrow c_{ij} + 1$
        $\Delta_{ij} \leftarrow \Delta_{ij} + (\hat{y}_{ti} - z_t)^2 - (\hat{y}_{tj} - z_t)^2$
        if $c_{ij} = m$ **then**
          $R \leftarrow R \setminus \{I\}$ where $I = i$ if $\Delta_{ij} > 0$ and $I = j$ otherwise.
        end if
      end for
    end if
  end if
end for

Theorem 2 *The* `noisy union` *algorithm is a KWIK algorithm with a KWIK bound of*

$$O\left(\frac{k}{\epsilon^2} \ln \frac{k}{\delta}\right) + \sum_{i=1}^{k} B_i\left(\frac{\epsilon}{8}, \frac{\delta}{k+1}\right),$$

when the parameter m is set appropriately:

$$m = \left\lceil \frac{128}{\epsilon^2} \ln \frac{k^2}{\delta} \right\rceil = O\left(\frac{1}{\epsilon^2} \ln \frac{k}{\delta}\right).$$

7 Case studies in reinforcement learning

This section identifies a number of KWIK-learnable environments for reinforcement-learning (RL), each of which results in a sample-efficient algorithm for the corresponding class of environments. We first introduce the basic notation, define the way we measure sample complexity in online RL, and then examines several prominent cases from the literature.

7.1 Markov decision processes and reinforcement learning

In reinforcement learning (Sutton and Barto 1998), a learner tries to learn to maximize the total reward it receives from an unknown environment through interaction. In the RL literature, the environment is often modeled as a *Markov decision process* or *MDP* (Puterman 1994), which is defined as a tuple: $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where: $\mathcal{S}$ is the *state space*, $\mathcal{A}$ is the *action space*, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{P}_{\mathcal{S}}$ is the unknown *transition function*, $R : \mathcal{S} \times \mathcal{A} \to \Re$ is the unknown *reward function*, and $\gamma \in (0, 1)$ is the *discount factor*. Here, $\mathcal{P}_{\mathcal{S}}$ denotes the set of probability distributions over $\mathcal{S}$. If $\mathcal{S}$ is countable, we may define $T(\cdot \mid s, a)$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$ as a probability *mass* function, so that $T(s' \mid s, a)$ is understood to be the probability of reaching the next state s' if action a is executed in state s. If $\mathcal{S}$ is uncountable, $T(\cdot \mid s, a)$ is understood to be a probability *density* function. In most practical situations, we may assume, without loss of generality, that R is a bounded function; namely, $R(s, a) \in [0, 1]$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. An MDP is called *finite* if both the state and action spaces are finite sets.

A *stationary policy* is a mapping from states to actions: $\pi : \mathcal{S} \to \mathcal{A}$. Given a policy π, we define the state-value function,[5] $V^{\pi}(s)$, as the expected cumulative discounted reward received by executing π starting from state s: $V^{\pi}(s) = \mathbf{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$, where r_t is the t-th reward obtained when following π from s. Similarly, the state–action value function, $Q^{\pi}(s, a)$, is the expected cumulative reward received by taking action a in state s and following π thereafter. To maximize the total rewards received from the environment, the agent desires an optimal policy π^* whose value functions, denoted by $V^*(s)$ and $Q^*(s, a)$, respectively, satisfy the conditions: $V^* = \max_{\pi} V^{\pi}$ and $Q^* = \max_{\pi} Q^{\pi}$.

Given the full model of an MDP (i.e., the five-tuple M), a set of standard algorithms exists for finding the optimal value function as well as the optimal policy, including linear programming, value iteration, and policy iteration (Puterman 1994). However, if the transition and/or reward functions are unknown, the decision maker has to gather information by interacting with the environment.

In a general reinforcement-learning setting which we call online RL, the agent chooses actions in discrete timesteps $t = 1, 2, 3, \ldots$. At every timestep t, the agent occupies a single state, denoted s_t, and has to choose an action $a_t \in \mathcal{A}$ according to some policy, and observes a sampled transition: the next state it occupies is drawn from the transition function, $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, and the immediate reward it receives is $r_t = R(s_t, a_t)$. It is straightforward to generalize our results to the case where rewards are stochastic. We call the tuple $\langle s_t, a_t, r_t, s_{t+1} \rangle$ a *transition*, which can be used as a training example for learning the reward and transition functions of the MDP. The learning agent may update its policy upon observing this transition. Therefore, an online RL algorithm can be viewed as a *nonstationary policy* that maps the sequence of transitions it observes so far to an action-selection rule. In contrast to stationary policies, nonstationary policies choose actions not just based on the current state, but also on previously experienced states, actions, and rewards.

In the rest of this section, we focus on a class of RL algorithms known as model-based approaches (see, *e.g.*, Moore and Atkeson 1993). These algorithms essentially learn the transition and reward functions of the underlying MDP, based on observed transitions, and then

[5] For MDPs with infinite (*e.g.*, continuous) state or action spaces, certain measure-theoretic assumptions are needed to guarantee the quantities we use, such as value functions and the Bellman operator (defined in Appendix C), are well-defined. This is beyond the scope of our paper, and interested readers are referred to Bertsekas and Shreve (1978) for details. Such assumptions are made implicitly in the rest of our paper.

use MDP solutions such as dynamic programming to compute an optimal policy of the approximate MDP $\hat{M}$. It is known that (through the various simulation lemmas in the literature such as Kearns and Singh (2002) as well as Lemma 9 in Appendix C) if the transition and reward functions of $\hat{M}$ are close to those of M, the optimal policy of $\hat{M}$ will be near-optimal in M. In model-based RL algorithms, therefore, the main challenge in learning is often that of learning the true MDP's transition and reward functions.

7.2 Model-based PAC-MDP reinforcement learning

Building on Kakade (2003)'s definition of *sample complexity of exploration*, Strehl et al. (2006a) formalizes the notion of *PAC-MDP* that can be viewed as an RL analogue to the PAC framework for supervised learning. Here, an RL algorithm is viewed as a non-stationary policy that takes actions based on history (including previously visited states and received rewards) and the current state. The number of timesteps during which this non-stationary policy is not near-optimal in the whole execution is called the algorithm's *sample complexity of exploration* (Kakade 2003). An algorithm is called PAC-MDP if its sample complexity is polynomial in relevant quantities, including the parameters that define the description complexity of the MDP, with high probability.

At the core of all existing model-based PAC-MDP algorithms such as `Rmax` (Brafman and Tennenholtz 2002; Kakade 2003) lies the idea of distinguishing between *known states*—states where the transition distribution and rewards can be accurately inferred from observed transitions—and *unknown states*. An important observation is that, if a class of MDPs can be KWIK-learned, then there exists an `Rmax`-style algorithm that is PAC-MDP for this class of MDPs. Theorem 3 formalizes this observation. The proof (in Appendix C) is through construction of a PAC-MDP algorithm called `KWIK-Rmax` that generalizes `Rmax` from finite MDPs to arbitrary MDPs. The proof relies on a form of the *simulation lemma* (Lemma 9), which relates value function approximation error to model approximation error, and on a generic PAC-MDP theorem given by Strehl et al. (2006a).[6]

Definition 2 Fix the state space $\mathcal{S}$, action space $\mathcal{A}$, and discount factor γ.

1. Define $\mathcal{X} = \mathcal{S} \times \mathcal{A}$, $\mathcal{Y}_T = \mathcal{P}_\mathcal{S}$, and $\mathcal{Z}_T = \mathcal{S}$. Let $\mathcal{H}_T \subseteq (\mathcal{X} \to \mathcal{Y}_T)$ be a set of transition functions of an MDP. $\mathcal{H}_T$ is *(efficiently) KWIK-learnable* if in the accuracy requirement of Definition 1, $|\hat{T}(\cdot \mid s,a) - T(\cdot \mid s,a)|$ is interpreted as the ℓ_1 distance:
$$|\hat{T}(\cdot \mid s,a) - T(\cdot \mid s,a)| := \begin{cases} \sum_{s' \in \mathcal{S}} |\hat{T}(s' \mid s,a) - T(s' \mid s,a)| & \text{if } \mathcal{S} \text{ is countable,} \\ \int_{s' \in \mathcal{S}} |\hat{T}(s' \mid s,a) - T(s' \mid s,a)| ds' & \text{otherwise.} \end{cases}$$
2. Define $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ and $\mathcal{Y}_R = \mathcal{Z}_R = [0,1]$. Let $\mathcal{H}_R \subseteq (\mathcal{X} \to \mathcal{Y}_R)$ be a set of reward functions of an MDP. $\mathcal{H}_R$ is *(efficiently) KWIK-learnable* if in the accuracy requirement of Definition 1, $|\hat{R}(s,a) - R(s,a)|$ is interpreted as the absolute value.
3. Let $\mathcal{M} = \{M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle \mid T \in \mathcal{H}_T, R \in \mathcal{H}_R\}$ be a class of MDPs. $\mathcal{M}$ is *(efficiently) KWIK-learnable* if both $\mathcal{H}_T$ and $\mathcal{H}_R$ are (efficiently) KWIK-learnable.

Theorem 3 *Let $\mathcal{M}$ be a class of MDPs with state space $\mathcal{S}$ and action space $\mathcal{A}$. If $\mathcal{M}$ can be (efficiently) KWIK-learned by algorithms $\mathbf{A}_T$ (for transition functions) and $\mathbf{A}_R$ (for reward*

[6] A slightly improved version is given in Theorem 21 of Li (2009), where the term $\ln 1/\delta$ is additive instead of multiplicative (as in Theorem 3).

functions) *with respective KWIK bounds* B_T *and* B_R, *then the* `KWIK-Rmax` *algorithm is PAC-MDP. In particular, if the following parameters are used*,

$$\epsilon_T = \frac{\epsilon(1-\gamma)^2}{16}, \qquad \epsilon_R = \frac{\epsilon(1-\gamma)}{16}, \qquad \epsilon_P = \frac{\epsilon(1-\gamma)}{24}, \qquad \delta_T = \delta_R = \frac{\delta}{4},$$

then the sample complexity of exploration of `KWIK-Rmax` *is*

$$O\left(\frac{B_T(\epsilon(1-\gamma)^2, \delta) + B_R(\epsilon(1-\gamma), \delta)}{\epsilon(1-\gamma)^2} \ln\frac{1}{\delta} \ln\frac{1}{\epsilon(1-\gamma)}\right).$$

The algorithm `KWIK-Rmax` (Algorithm 3) relies on two KWIK algorithms, $\mathbf{A}_T$ (using parameters ϵ_T, δ_T) and $\mathbf{A}_R$ (using parameters ϵ_R, δ_R), for KWIK-learning the MDP's transition and reward functions, respectively, and maintains an estimate of the MDP called the *empirical known state–action MDP* (Strehl et al. 2006a). The estimate distinguishes two types of state–actions: for state–actions where both $\mathbf{A}_T$ and $\mathbf{A}_R$ can make valid predictions, the predictions must be accurate with high probability (the accuracy requirement for KWIK algorithms) and thus their dynamics are *known*; for other state–actions, their transition and reward functions cannot be accurately estimated and thus they are *unknown*. By assigning the largest possible reward (which is 1) to all unknown state–actions and let them self-loop, the agent is encouraged to explore these state–actions unless the probability of reaching them is too small. When the probability of reaching unknown states is small, we can show near-optimality of the greedy policy. Since the number of visits to an unknown state–action is polynomial in relevant quantities (the sample complexity requirement for KWIK algorithms), the number of timesteps the algorithm does not behave near-optimally is also a

Algorithm 3 `KWIK-Rmax`

Inputs: $\mathcal{S}$, $\mathcal{A}$, γ, $\mathbf{A}_T$ (with parameters ϵ_T, δ_T), $\mathbf{A}_R$ (with parameters ϵ_R, δ_R), ϵ_P
for all timesteps $t = 1, 2, 3, \ldots$ **do**
  // Update the empirical known state–action MDP $\hat{M} = \langle \mathcal{S}, \mathcal{A}, \hat{T}, \hat{R}, \gamma \rangle$
  for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    if $\mathbf{A}_T(s, a) = \perp$ or $\mathbf{A}_R(s, a) = \perp$ **then**
      $\hat{T}(s'|s, a) = \begin{cases} 1 & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases}$ and $\hat{R}(s, a) = 1$.
    else
      $\hat{T}(\cdot|s, a) = \mathbf{A}_T(s, a)$ and $\hat{R}(s, a) = \mathbf{A}_R(s, a)$.
    end if
  end for
  Compute a near-optimal value function Q_t of $\hat{M}$ such that $|Q_t(s, a) - Q^*_{\hat{M}}(s, a)| \leq \epsilon_P$ for all (s, a), where $Q^*_{\hat{M}}$ is the optimal state–action value function of $\hat{M}$.
  Observe the current state s_t, take action $a_t = \arg\max_{a \in \mathcal{A}} Q_t(s_t, a)$, receive reward r_t, and transition to the next state s_{t+1}.
  if $\mathbf{A}_T(s_t, a_t) = \perp$ **then**
    Inform $\mathbf{A}_T$ of the sample $(s_t, a_t) \to s_{t+1}$.
  end if
  if $\mathbf{A}_R(s_t, a_t) = \perp$ **then**
    Inform $\mathbf{A}_R$ of the sample $s_t \to r_t$.
  end if
end for

polynomial. This intuition is originally from Rmax (Brafman and Tennenholtz 2002) and the sample-complexity proof of Kakade (2003), and is also the basic idea of our PAC-MDP proof in Appendix C.

A few caveats are in order regarding practical issues when implementing Algorithm 3:

1. The definitions of $\hat{T}$ and $\hat{R}$ are conceptual rather than operational. For finite MDPs, one may represent $\hat{T}$ by a matrix of size $O(|\mathcal{S}|^2|A|)$ and $\hat{R}$ by a vector of size $O(|\mathcal{S}||\mathcal{A}|)$. For structured MDPs, more compact representations are possible. For instance, MDPs for some linear dynamical systems can be represented by matrices of finite dimension (Sects. 7.5 and 7.6), and factored MDPs can be represented by a dynamic Bayes net (Sect. 7.7).
2. It is unnecessary to update $\hat{T}$ and $\hat{R}$ and recompute Q_t for every timestep t. The known state–action MDP $\hat{M}$ (and thus $Q^*_{\hat{M}}$ and Q_t) remains unchanged unless some unknown state–action becomes known. Therefore, one may update $\hat{M}$ and Q_t only when $\mathbf{A}_T$ or $\mathbf{A}_R$ obtain new samples in lines 13 or 16.
3. It is unnecessary to compute Q_t for all (s, a). In fact, it suffices to guarantee that Q_t is ϵ_P-accurate in state s_t: $|Q_t(s_t, a) - Q^*_{\hat{M}}(s_t, a)| < \epsilon_P$ for all $a \in \mathcal{A}$. This kind of *local* planning often requires less computation than *global* planning.
4. Given the approximate MDP $\hat{M}$ and the current state s_t, the algorithm computes a near-optimal action for s_t. This step can be done efficiently using dynamic programming for finite MDPs. In general, however, doing so is computationally expensive (Chow and Tsitsiklis 1989). Fortunately, recent advances in approximate local planning have made it possible for large-scale problems (Kearns et al. 2002; Kocsis and Szepesvári 2006).

After explaining why existing computational learning models, PAC and MB, are not adequate for sample-efficient reinforcement learning in Sect. 7.3, the remaining subsections consider various subclasses of MDPs and unify most existing PAC-MDP algorithms using the KWIK framework developed in previous sections. Section 7.4 considers finite MDPs without considering generalization across states, while Sects. 7.5–7.7 show how generalization can be combined with KWIK to make use of structural assumptions of corresponding MDP classes to obtain more efficient RL algorithms. Finally, Sect. 7.8 shows how we may obtain a PAC-MDP algorithm for factored-state MDPs with unknown factorization structures by streamlining four KWIK algorithms we have developed, resulting in a novel algorithm that is significantly better than the state-of-the-art result. In all these cases, we focus on learning the transition function and assume the reward function is known. The extension to KWIK-learning reward functions is straightforward. The results are summarized in Table 1.

To simplify exposition and focus on application of KWIK in MDP learning, we avoid direct analysis of how to KWIK-learn the transition functions in terms of the ℓ_1 distance in Sects. 7.5 and 7.6, as required by Theorem 3. However, the discrepancy functions we use in those cases are both polynomially equivalent to the ℓ_1 distance. More details are found in the original papers (Strehl and Littman 2008; Brunskill et al. 2009).

7.3 The inadequacies of PAC and MB for RL

Given the relation between KWIK and PAC-MDP RL in the previous subsection, a natural question is whether the existing computational learning models such as PAC and MB are sufficient for MDP learning in reinforcement learning. We show that in general they are not. The argument against PAC stems from its distributional assumption: PAC assumes independently, identically distributed samples, which is clearly not true in online RL except quite restricted cases. Instead, the agent's shifting policy determines the samples it sees.

Table 1 KWIK bounds and the component KWIK algorithms for a number of prominent MDP classes. Refer to the text for definitions of the quantities in the KWIK bounds

Case	Subalgorithms		KWIK Bound
Finite		`input partition`	$O(\frac{n^2m}{\epsilon^2}\ln\frac{nm}{\delta})$
	←	`dice learning`	
Linear		`output combination`	$\tilde{O}(\frac{n_S^2 n}{\epsilon^4})$
Dynamics	←	`noisy linear regression`	
Normal Offset		`input partition`	$O(\frac{n_\tau^2 m^2 n^8}{\epsilon^4\delta}\ln\frac{n_\tau mn}{\delta})$
	←	`output combination`	
	←	`coin learning`	
Factored with		`cross product`	$O(\frac{n^3 mDN^{D+1}}{\epsilon^2}\ln\frac{nmN}{\delta})$
Known $\mathcal{P}$	←	`input partition`	
	←	`dice learning`	
Factored with		`output combination`	$O(\frac{n^{D+3}mDN^{D+1}}{\epsilon^2}\ln\frac{nmN}{\delta})$
Unknown $\mathcal{P}$	←	`noisy union`	
	←	`input partition`	
	←	`dice learning`	

This policy-driven sampling of state–actions is most conveniently viewed in the adversarial setting, which is outside the scope of PAC, but within the realm of MB.

MB algorithms, however, are not necessarily capable of on-demand labeling of state–actions as known or unknown, a pivotal requirement for existing PAC-MDP algorithms. For instance, consider an exponentially large MDP where each state corresponds to a configuration of n Boolean variables and rewards are determined by a single conjunction over the current variable configuration. More specifically, there are two actions a_1, which flips the rightmost bit, and a_2, which shifts the bits left. If the state's configuration satisfies the unknown conjunction, either action produces a reward of 1, otherwise the reward is -1. Assume, for the sake of argument, the transition function is known. One can learn about the reward function by visiting all 2^n states, but an MB algorithm that defaults to "false" (*c.f.*, Sect. 3.2) can learn conjunctions with only $n+1$ mistakes, so it is tempting to adopt an MB perspective in learning the rewards. But this will not translate directly to a PAC-MDP algorithm because there is no basis for guiding exploration. At any time, states that have actually been experienced as "false" (-1 reward) and those that have never been explored (unknown reward) cannot be distinguished. They are both simply labeled "false". Thus, the agent could end up revisiting ("exploring") the same "false" state over and over, expecting a different reward. One could start labeling the states that have been explored, but this is by definition a KWIK algorithm!

In contrast to MB and PAC, KWIK embodies the conditions sufficient for the model learning component of a model-based PAC-MDP algorithm. The KWIK framework harbors no distributional assumptions, and explicitly designates the known and unknown regions of the state/action space. As we saw in Sect. 3.2, this limits the models that can be efficiently learned to a proper subset of those that can be learned by MB, but we will show in the remaining subsections that this set still covers a large number of natural and interesting domain types from the reinforcement-learning literature.

7.4 KWIK-learning finite MDPs

Finite MDPs have received the bulk of the attention in the RL literature because of their mathematical simplicity. A finite MDP $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ consists of $n = |\mathcal{S}|$ states and $m = |\mathcal{A}|$ actions. For each combination of state (s) and action (a) and next state (s'), the transition function returns a probability, denoted $T(s' \mid s, a)$. As the reinforcement-learning agent moves around in the state space, it observes state–action–next-state transitions and must predict the probabilities for transitions it has not yet observed. In the model-based setting, an algorithm learns a mapping from the size nm input set of state–action combinations to multinomial distributions over the next states via multinomial observations. Thus, the problem can be solved via `input partition` over a set of individual probabilities learned via `dice learning`. The resulting KWIK bound is

$$B(\epsilon, \delta) = O\left(\frac{n^2 m}{\epsilon^2} \ln \frac{nm}{\delta}\right).$$

This approach is precisely what is found in most sample-efficient RL algorithms in the literature (Kearns and Singh 2002; Brafman and Tennenholtz 2002; Kakade 2003; Strehl et al. 2006a). More recently, Szita and Szepesvári (2010) propose a variant of `Rmax` and a refined analysis that results in a linearithmic dependence on n, matching the best previous bound (*c.f.*, Sect. 7.9).

We note that a similar idea can be applied to MDPs with infinite state and action spaces, provided that the MDPs satisfy the *local modeling* assumption (Kakade et al. 2003). Intuitively, the dependence on sampling each state–action m times in the finite case is replaced by a requirement of gathering some m samples for each action in each partition of the space induced by the local modeling assumption. The KWIK bound for learning this type of MDPs is in general dependent on the smallest number of partitions (a.k.a. the *covering number*), rather than $|\mathcal{S}|$.

7.5 KWIK-learning continuous MDPs with linear dynamics

In many robotics and control applications, the systems being manipulated possess infinite state spaces and action spaces, but their dynamics are governed by a system of linear equations (see, *e.g.*, Sontag 1998). Our model formulation is based on Strehl and Littman (2008), and is slightly different from Abbeel and Ng (2005). Here, $\mathcal{S} \subseteq \Re^{n_S}$ and $\mathcal{A} \subseteq \Re^{n_A}$ are the state and action spaces, respectively. The transition function T is a multivariate normal distribution:

$$T(\cdot \mid s, a) = \mathcal{N}\big(F\phi(s, a), \sigma^2 I\big),$$

where $\phi : \Re^{n_S + n_A} \to \Re^n$ is a basis function satisfying $\|\phi(\cdot, \cdot)\| \leq 1$, $F \in \Re^{n_S \times n}$ is a matrix, σ^2 is some positive number, and $I \in \Re^{n_S \times n_S}$ is the identity matrix. We assume ϕ and σ^2 are given, but F is unknown.

For such linearly parameterized transition functions, the expectation of each component of the next state is linear in the feature of the current state–action, and the coefficients correspond to the numbers in the corresponding rows in F. If we define the discrepancy metric in $\Re^{n_S}$ as the ℓ_2 norm, then the transition function of the class of linearly parameterized MDPs can be KWIK-learned via `output combination` over the n_S state components using $\alpha = 1/\sqrt{n_S}$, each of which can be KWIK-learned by `noisy linear regression`.

The resulting KWIK bound is

$$B(\epsilon, \delta) = \tilde{O}\left(\frac{n_S^2 n}{\epsilon^4}\right).$$

7.6 KWIK-learning typed MDPs with normal offset dynamics

The linear transition model above is a class of parameterized transition functions for continuous MDPs. A related class is called *type-conditional dynamics* (see, *e.g.*, Leffler et al. 2007), where the transition functions are determined by a *type* $\tau(s)$ assigned to each state s and action a.

Here, we consider a specific subset of this class, adopted from Brunskill et al. (2008), where $\mathcal{S} \subseteq \Re^n$ and the next-state distribution is given by the following multivariate normal distribution:

$$T(\cdot \mid s, a) \sim \mathcal{N}\big(s + \mu_{\tau(s)a}, \Sigma_{\tau(s)a}\big),$$

where $\mu_{\tau(s)a} \in \Re^n$ and $\Sigma_{\tau(s)a} \in \Re^{n \times n}$ are the mean and covariance matrix of the normal distribution for the type–action pair $(\tau(s), a)$. In other words, the *offset* of states is normally distributed. We assume the number of actions, denoted $m = |\mathcal{A}|$, and the number of types, denoted $n_\tau = |\{\tau(s) \mid s \in \mathcal{S}\}|$, are both finite.

An example domain where such dynamics may arise is robot navigation across varying terrain. The distribution of the offset of the robot's position after taking an action (such as turn-left or go-forward) depends on the this action as well as the type of the terrain (such as sand, wood, or ice, etc.). In many real-world applications, the number of types is often small although the number of states can be astronomically large or even infinite. Typed offset dynamics, therefore, provide a compact way to represent the MDP. More motivations are found in Brunskill et al. (2009).

For each type–action (τ, a), the components in $\mu_{\tau a}$ and $\Sigma_{\tau a}$ can be interpreted as means of observations. For instance, assume that we have acquired a sample transition, (s, a, r, s'), then by definition

$$\mu_{\tau(s)a}[i] = \mathbf{E}_{s' \sim T(\cdot|s,a)}\big[s'[i] - s[i]\big]$$

for all $i = 1, 2, \ldots, n$. Therefore, we may decompose the problem of KWIK-learning the offset means using `input partition` over all type–actions. For each (τ, a) pair, learning $\mu_{\tau a}$ is turned into one of KWIK-learning the mean of n random variables via `output combination` using $\alpha = 1$, each of which can be solved by `coin learning`.[7] A similar reduction can be applied to KWIK-learn the covariance matrix $\Sigma_{\tau a}$. To ensure the total variance between the true Gaussian distribution, $\mathcal{N}(\mu_{\tau a}, \Sigma_{\tau a})$, and the estimated distribution, $\mathcal{N}(\hat{\mu}_{\tau,a}, \hat{\Sigma}_{\tau,a})$, is at most ϵ, the KWIK bound is

$$B(\epsilon, \delta) = \tilde{O}\left(\frac{n_\tau^2 m^2 n^8}{\lambda_{\min}^2 \epsilon^4 \delta}\right),$$

where $\lambda_{\min}$ is the smallest singular value of the covariance matrix, $\Sigma_{\tau,a}$. A full derivation of this KWIK bound is provided in Li (2009).

[7] Since a normally distributed random variable is unbounded, some tricks are necessary. Details are found in Brunskill et al. (2009).

7.7 KWIK-learning factored-state MDPs with known structure

In many applications, the MDP's state space is most naturally represented as the cross product of subspaces, each of which corresponds to a state variable. Under such conditions, dynamic programming and reinforcement learning often face the "curse of dimensionality" (Bellman 1957), which says that the number of states in the MDP is exponential in the number of state variables, rendering most finite MDP-based learning/optimization algorithms intractable. Factored-state representations (Boutilier et al. 1999) are compact representations of MDPs that avoid explicitly enumerating all states when defining an MDP. Although planning in factored-state MDPs remains hard in the worst case, their compact structure can reduce the sample complexity of learning significantly since the number of parameters needed to specify a factored-state MDP can be exponentially smaller than in the unstructured case.

In a factored-state MDP, let $m = |\mathcal{A}|$ be the number of actions; every state is a vector consisting of n components: $s = (s[1], s[2], \ldots, s[n]) \in \mathcal{S}$. Each component $s[i]$ is called a *state variable* and can take values in a finite set $\mathcal{S}_i$. The whole state space $\mathcal{S}$ is thus $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. Without loss of generality, assume $N = |\mathcal{S}_i|$ for all i and thus $|\mathcal{S}| = N^n$. The transition function is factored into the product of n transition functions, one for each state variable:

$$T(s' \mid s, a) = \prod_{i=1}^{n} T_i(s'[i] \mid s, a).$$

In other words, the values of all the next-state variables are independent of each other, conditioned on the current state–action (s, a). Note that one can also define a DBN with "synchronous" edges, denoting a dependency between factor values at the same timestep. For simplicity, we do not consider such dependencies in our analysis, but the results are easily extended to this case by (at most) doubling the number of factors being considered as possible parents (though not the actual number of parents for any given node). Furthermore, we assume that the distribution $T_i(\cdot \mid s, a)$ depends on a small subset of $\{s[1], s[2], \ldots, s[n]\}$, denoted $\mathcal{P}(i)$. This assumption is valid in many real-life applications. Let D be the largest size of $\mathcal{P}_i$: $D = \max_i |\mathcal{P}_i|$. Although we treat D as a constant, we include D explicitly in the KWIK bounds to show how D affects learning efficiency.

Using the quantities defined above, the transition function can be rewritten as

$$T(s' \mid s, a) = \prod_{i=1}^{n} T_i(s'[i] \mid \mathcal{P}(i), a).$$

An advantage of this succinct representation is that, instead of representing the complete conditional probability table, $T_i(\cdot \mid s, a)$, which has $N^n m$ entries, we only need to use the smaller $T_i(\cdot \mid \mathcal{P}(i), a)$, which has at most $N^D m$ entries. If $D \ll n$, we are able to achieve an exponentially more compact representation. This kind of transition functions can be represented as *dynamic Bayesian networks* or *DBNs* (Dean and Kanazawa 1989). Here, we consider the case where the structure $(\mathcal{P}(i))$ is known *a priori* and show how to relax this assumption in Sect. 7.8.

The reward function can be represented in a similar way as the sum of *local reward functions*. We assume, for simplicity, that $R(s, a)$ is known and focus on KWIK-learning the transition functions. Our algorithm and insights still apply when the reward function is unknown.

Transitions in a factored-state MDP can be thought of as mappings from vectors $(s[1], s[2], \ldots, s[n], a)$ to vectors $(s'[1], s'[2], \ldots, s'[n])$. Given known dependencies, `cross product` with $\alpha = 1/n$ can be used to learn each component of the transition function.[8] Each component is, itself, an instance of `input partition` applied to `dice learning`. This three-level KWIK algorithm provides an approach to learn the transition function of a factored-state MDP with the following KWIK bound:

$$B(\epsilon, \delta) = O\left(\frac{n^3 m D N^{D+1}}{\epsilon^2} \ln \frac{nmN}{\delta}\right).$$

This insight can be used to derive the factored-state-MDP learning algorithm used by Kearns and Koller (1999).

7.8 KWIK-learning factored-state MDPs with unknown structures

Without known structural dependencies of the DBN, learning a factored-state MDP is more challenging. Strehl et al. (2007) showed that each possible dependence structure can be viewed as a separate hypothesis and provided an algorithm for learning the dependencies in a factored-state MDP while learning the transition probabilities. The resulting KWIK bound is super-quadratic in $k = \Theta(2^D)$, where D, as before, is the maximum in-degree of the true DBN. We can construct a conceptually simpler algorithm for this problem using components introduced throughout this paper. As a whole, the algorithm can be viewed as a four-level KWIK algorithm with a `cross product` at the top to decompose the transitions for the separate components of the factored-state representation. Each of these n individual factor transitions is learned using a separate copy of the `noisy union` algorithm. Within each of these, a union is performed over the $\binom{n}{D}$ possible parent configurations for the given state component. As in the "known structure" case, an `input partition` algorithm is used for each of those possible configurations to handle the different combinations of parent values and action, and finally `dice learning` is used to learn the individual transition probabilities themselves. The resulting KWIK bound is

$$B(\epsilon, \delta) = O\left(\frac{n^{D+3} m D N^{D+1}}{\epsilon^2} \ln \frac{nmN}{\delta}\right).$$

Note that our `noisy union` component is conceptually simpler, significantly more efficient ($k \ln k$ vs. $k^2 \ln k$ dependence on $k = \binom{n}{D}$), and more generally applicable than the similar procedure employed by Strehl et al. (2007). Thus, our improvement is exponential in D and polynomial in n. Preliminary studies show that the new algorithm works much better in practice (Diuk et al. 2009).

7.9 Model-free PAC-MDP reinforcement learning

The previous subsections discuss how to create model-based PAC-MDP RL algorithms with help of KWIK algorithms for various MDP subclasses. While model-based algorithms often have better sample complexity in practice, they usually maintain an implicit model of the unknown MDP and have to repeatedly solve this approximate MDP (such as in `KWIK-Rmax`) to obtain a value function or policy for taking actions.[9] Despite recent advances

[8] Here, this value of α suffices to guarantee that the combined transition distribution differs from the true transition distribution by ϵ in terms of ℓ_1-distance; see Li (2009) for a proof.

[9] For finite MDPs, it is possible to replace exact planning by incremental planning (Strehl et al. 2006a), which is much more efficient.

in approximate planning in MDPs (see, *e.g.*, Kocsis and Szepesvári 2006), solving MDPs remains challenging in general.

In contrast, *model-free* algorithms directly learn optimal value functions, from which a greedy policy can be easily computed. These algorithms are therefore much more efficient in terms of both time and space. The first model-free PAC-MDP algorithm for finite MDPs, known as `delayed Q-learning` (Strehl et al. 2006b, 2009), is a variant of `Q-learning` that carefully maintains an optimistic value function and takes greedy actions in all timesteps. The algorithm requires only $\Theta(nm)$ space complexity and $O(\ln m)$ per-step time complexity to achieve a sample complexity of $\tilde{O}(nm/\epsilon^4(1-\gamma)^8)$, where n and m are the numbers of states and actions. Recently, it was shown that the dependence on n is optimal (Li 2009; Strehl et al. 2009). In contrast, the best existing computational complexity is $O(n \ln m)$ (Strehl et al. 2006a).

Despite the nice computational advantages of model-free algorithms, these algorithms are unfortunately harder to analyze. In a recent work by Li and Littman (2010), a finite-horizon reinforcement learning problem is reduced to a series of KWIK regression problems so that we can obtain a PAC-MDP model-free RL algorithm as long as the individual KWIK regression problems are solvable. However, it remains unclear how to devise a KWIK-based model-free RL algorithm in discounted problems without first converting it into a finite-horizon one.

8 Conclusion and future work

We described the KWIK ("knows what it knows") model of supervised learning, which identifies and generalizes a key model learning component common to a class of algorithms for efficient exploration. We provided algorithms for a set of basic hypothesis classes given deterministic and noisy observations as well as methods for composing hypothesis classes to create more complex algorithms. One example algorithm consisted of a four-level decomposition of an existing learning algorithm from the reinforcement-learning literature.

By providing a set of example algorithms and composition rules, we hope to encourage the use of KWIK algorithms as a component in machine-learning applications as well as spur the development of novel algorithms. Such a direction has seen promising results. Brunskill et al. (2009) successfully applied the algorithm in Sect. 7.6 to a robot navigation problem. More recently, an associative reinforcement learning algorithm motivated by the KWIK algorithm of Walsh et al. (2009) has showed encouraging results in a challenging online news article recommendation problem on Yahoo! Frontpage (Li et al. 2010).

The study in the present paper raises several important theoretical directions. We list seven of them to conclude the paper. First, we would like to extend the KWIK framework to the "unrealizable" case by removing the assumption that the hypothesis class $\mathcal{H}$ contains the true hypothesis h^*. This is sometimes called *agnostic learning* (Kearns et al. 1994). We are able to provide a general-purpose KWIK algorithm when $\mathcal{H}$ is finite, resulting in a KWIK bound of $O(|\mathcal{H}|)$,[10] but it remains open how to handle unrealizable target functions with infinite-size hypothesis classes.

Second, it is not clear how to characterize the dimension, $\dim(\mathcal{H})$, of a hypothesis class $\mathcal{H}$ in a way that can be used to derive KWIK bounds. For finite hypothesis classes, the

[10] This bound can be achieved by a variation of `enumeration`. In this variant, $\hat{\mathcal{H}}$ is the set of hypotheses that have been within ϵ-accuracy of the observed outputs, and the algorithm returns $\perp$ if the range of $\hat{L}$ is larger than 2ϵ and otherwise it returns the midpoint of this set.

cardinality of $\mathcal{H}$ is a reasonable choice, as is used in `enumeration`, but a more general definition is needed for infinite hypothesis classes. Similar efforts in the concept-learning literature (Angluin 2004) may provide useful insights for this problem.

Third, it is important to find a general scheme for taking a KWIK algorithm for a deterministic class and updating it to work in the presence of noise. In the n-dimensional linear functions learning problem, we have seen that the KWIK bound grows from $O(n)$ to $\tilde{O}(n^3)$ by the algorithms of Strehl and Littman (2008) and Walsh et al. (2009). In general, we expect the KWIK bound to increase significantly when noise is present unless the environment is oblivious (Cesa-Bianchi et al. 2009).

Fourth, we would like to explore the relationship between KWIK and other online learning models. For instance, the selective sampling and label efficient prediction models share some similarity with KWIK and we might be able to modify algorithms in these frameworks to satisfy the KWIK criteria.

Fifth, this paper only considers linear approximation architectures. It is possible that nonlinear architectures may be preferred in some cases given their greater representational power. A common approach to introducing nonlinearity in practice is to use a function that is linear in nonlinear features, but a direct analysis of nonlinear architectures in KWIK could be useful.

Finally, it is of theoretical as well as practical interest to utilize prior information in KWIK. Prior information may be, for instance, a prior distribution of $h^* \in \mathcal{H}$. Ideally, a KWIK algorithm that makes use of such prior information should still satisfy the requirements in Definition 1, but may enjoy a significantly smaller KWIK bound by leveraging an informative prior.

Acknowledgements This material is based in part upon work supported by the National Science Foundation under Grant No. NSF IIS-0325281 and a contract from DARPA IPTO. We thank Roni Khardon and the anonymous reviewers for valuable comments that have greatly improved an earlier draft of this paper. We also thank István Szita for many helpful discussions, and Mark Kroon for pointing out a mistake in our original presentation of the algorithm in Sect. 7.8.

Appendix A: Proof of Theorem 1

A.1 Proof of Lemma 1

We define the random variables $Y_i = (z_i - \hat{\theta}^\mathsf{T} x_i)^2$ for $i = 1, \ldots, m$. Let $Y = \sum_{i=1}^m Y_i$. Note that $\mathbf{E}[Y_i \mid x_i] = \mathbf{Var}[z_i - \hat{\theta}^\mathsf{T} x_i \mid x_i] + \mathbf{E}[z_i - \hat{\theta}^\mathsf{T} x_i \mid x_i]^2 = \sigma_i^2 + [(\theta - \hat{\theta})^\mathsf{T} x_i]^2$. In the first step we used the identity $\mathbf{E}[X^2] = \mathbf{Var}[X] + \mathbf{E}[X]^2$ for random variable X. The second step follows from the assumption that $\mathbf{E}[z_i \mid x_i] = \theta^\mathsf{T} x_i$. We have shown that

$$\mathbf{E}[Y_i \mid x_i] = \sigma^2 + \left((\theta - \hat{\theta})^\mathsf{T} x_i\right)^2. \tag{4}$$

To bound the absolute value of Y_i, we use the identity (consequence of the Cauchy-Schwartz inequality) $|a^\mathsf{T} b| \leq \|a\| \cdot \|b\|$ for any two vectors a, b and our assumptions that $|z_i| \leq 1$, $\|x_i\| \leq 1$, and $\|\hat{\theta}\| \leq 1$ for all $i = 1, \ldots, m$. We have that $|Y_i| = (z_i - \hat{\theta}^\mathsf{T} x_i)^2 \leq$

$(1 + \|\hat{\theta}\|\|x_i\|)^2 \leq 4$. We then apply Hoeffding (1963)'s inequality[11] to yield:

$$\Pr(|Y - \mathbf{E}[Y]| \geq \alpha) \leq 2\exp\left(-\frac{\alpha^2}{8m}\right).$$

Here we use $\mathbf{E}[Y]$ to denote $\sum_{i=1}^{m} \mathbf{E}[Y_i \mid x_i]$. Setting the right-hand side of the above equation to be at most δ yields $\alpha \geq \sqrt{8m\ln(2/\delta)}$. We have shown that

$$\Pr\left(|Y - \mathbf{E}[Y]| \geq \sqrt{8m\ln(2/\delta)}\right) \leq \delta.$$

Now, by (4),

$$|Y - \mathbf{E}[Y]| = \left|Y - \sum_{i=1}^{m} \mathbf{E}[Y_i \mid x_i]\right| = \left|Y - \sum_{i=1}^{m} \sigma_i^2 - \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2\right|.$$

Combining these equations shows that

$$\left|Y - \sum_{i=1}^{m} \sigma_i^2 - \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2\right| < \sqrt{8m\ln(2/\delta)}$$

holds with high probability. This equation gives

$$-\sqrt{8m\ln(2/\delta)} < Y - \sum_{i=1}^{m} \sigma_i^2 - \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2.$$

The random variables Y_i model the squared errors for predictions of z_i given x_i using parameter vector $\hat{\theta}$. Similarly, we define random variables $Z_i = (z_i - \theta^{\mathsf{T}} x_i)^2$ and $Z = \sum_{i=1}^{m} Z_i$ that model the squared errors using the "correct" parameter vector θ. Using the previous argument we have that, with probability at least $1 - \delta$,

$$Z - \sum_{i=1}^{m} \sigma_i^2 \leq \sqrt{8m\ln(2/\delta)}.$$

We can rewrite the two previous equations to yield

$$Y > \sum_{i=1}^{m} \sigma_i^2 + \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2 - \sqrt{8m\ln(2/\delta)},$$

and

$$Z \leq \sum_{i=1}^{m} \sigma_i^2 + \sqrt{8m\ln(2/\delta)}.$$

[11] Hoeffding's bound is often applied to independent random variables. In our case, the random variables Y_i are not independent, however Y_i is conditionally independent of Y_j for all $j < i$ given x_i so the sequence $Y_i - \mathbf{E}[Y_i \mid x_i]$ for $i = 1, \ldots, m$ is a martingale difference sequence (with respect to a filtration based on the x_i) and thus Hoeffding's bound applies.

Hence, combining these two inequalities and applying the union bound, we have that

$$Y - Z > \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2 - 2\sqrt{8m \ln(2/\delta)},$$

holds with probability at least $1 - 2\delta$. Reorganizing terms and using the assumption in the lemma, we have

$$Y > Z + \sum_{i=1}^{m} \left((\theta - \hat{\theta})^{\mathsf{T}} x_i\right)^2 - 2\sqrt{8m \ln(2/\delta)} > Z + w = \sum_{i=1}^{m} (z_i - \theta^{\mathsf{T}} x_i)^2 + w.$$

A.2 Proof of Lemma 2

First, note that since the singular vectors $\{u_i\}$ (defined in Sect. 5) are orthonormal, x can be written as a linear combination of them by $x = \sum_{i=1}^{n} (u_i^{\mathsf{T}} x) u_i$. Now claim the following equality holds:

$$x = X^{\mathsf{T}} \bar{q} + \left(\sum_{i=k+1}^{n} u_i u_i^{\mathsf{T}} \right) x, \tag{5}$$

where $\bar{q} = X \bar{U} \bar{\Lambda}^{-1} \bar{U}^{\mathsf{T}} x$. To see this, we have to show $\sum_{i=1}^{k} (u_i^{\mathsf{T}} x) u_i = X^{\mathsf{T}} \bar{q}$:

$$\begin{aligned}
X^{\mathsf{T}} \bar{q} &= X^{\mathsf{T}} X \bar{U} \bar{\Lambda}^{-1} \bar{U}^{\mathsf{T}} x \\
&= U \Lambda U^{\mathsf{T}} \bar{U} \bar{\Lambda}^{-1} \bar{U}^{\mathsf{T}} x \\
&= \sum_{j=1}^{n} \left(\lambda_j u_j u_j^{\mathsf{T}}\right) \sum_{i=1}^{k} \left(\lambda_i^{-1} u_i u_i^{\mathsf{T}}\right) x \\
&= \sum_{j=1}^{n} \sum_{i=1}^{k} \left(\frac{\lambda_j}{\lambda_i} u_j u_j^{\mathsf{T}} u_i u_i^{\mathsf{T}} \right) x \\
&= \sum_{i=1}^{k} u_i u_i^{\mathsf{T}} x,
\end{aligned}$$

where the first step uses the definition of $\bar{q}$, the second uses singular value decomposition of $X^{\mathsf{T}} X$ (as defined in Sect. 5), the third uses the fact that Λ and $\bar{\Lambda}$ are diagonal matrices, the fourth is just changing the order of two summations, and the last uses the fact that $u_j^{\mathsf{T}} u_i$ is identity for $i = j$ and zero otherwise. Thus, we have proved (5).

Next, by two applications of the Cauchy-Schwartz inequality, we have that

$$\begin{aligned}
\left|(\theta - \hat{\theta})^{\mathsf{T}} x\right| &= \left| (\theta - \hat{\theta})^{\mathsf{T}} \left(X^{\mathsf{T}} \bar{q} + \left(\sum_{i=k+1}^{n} u_i u_i^{\mathsf{T}} \right) x \right) \right| \\
&\leq \|\bar{q}\| \cdot \|X(\theta - \hat{\theta})\| + \|\theta - \hat{\theta}\| \cdot \left\| \left(\sum_{i=k+1}^{n} u_i u_i^{\mathsf{T}} \right) x \right\|.
\end{aligned}$$

Since $\|\theta\| \leq 1$ and $\|\hat{\theta}\| \leq 1$, we have $\|\theta - \hat{\theta}\| \leq \|\theta\| + \|\hat{\theta}\| \leq 2$. Therefore, we have shown

$$\left|(\theta - \hat{\theta})^{\mathsf{T}} x\right| \leq \|\bar{q}\| \cdot \|X(\theta - \hat{\theta})\| + 2\left\|\left(\sum_{i=k+1}^{n} u_i u_i^{\mathsf{T}}\right) x\right\|.$$

For any matrix M, let $\|M\|$ denote the *spectral norm* of M (largest singular value of M). Note that $\|(\sum_{i=k+1}^{n} u_i u_i^{\mathsf{T}})x]\| = \|[u_{k+1} \cdots u_n][u_{k+1} \cdots u_n]^{\mathsf{T}} x\| \leq \|[u_{k+1} \cdots u_n]\| \cdot \|[u_{k+1} \cdots u_n]^{\mathsf{T}} x\| \leq \|[u_{k+1} \cdots u_n]^{\mathsf{T}} x\| = \|\bar{u}\|$. The second inequality results from the fact that the maximum singular value of the matrix $[u_{k+1} \cdots u_n]$ is at most 1. Using the fact that $\|X(\theta - \hat{\theta})\| = \Delta_{\mathrm{E}}(\hat{\theta})$ and combining the previous derivations yields the desired result.

A.3 Proof of Theorem 1

We will need the following lemma:

Lemma 3 *Suppose $x_i \in \Re^n$ satisfy $\|x_i\| \leq 1$ and $a_i \in [-1, 1]$ for $i = 1, \ldots, m$. Let $\theta_1, \theta_2 \in \Re^n$ be two vectors satisfying $\|\theta_1\| \leq 1$ and $\|\theta_2\| \leq 1$. Then,*

$$\left|\sum_{i=1}^{m} \left(a_i - \theta_1^{\mathsf{T}} x_i\right)^2 - \sum_{i=1}^{m} \left(a_i - \theta_2^{\mathsf{T}} x_i\right)^2\right| \leq 4m\|\theta_1 - \theta_2\|.$$

Proof

$$\begin{aligned}
&\left|\sum_{i=1}^{m} \left(a_i - \theta_1^{\mathsf{T}} x_i\right)^2 - \sum_{i=1}^{m} \left(a_i - \theta_2^{\mathsf{T}} x_i\right)^2\right| \\
&\quad = \left|\sum_{i=1}^{m} \left(\left(a_i - \theta_1^{\mathsf{T}} x_i\right)^2 - \left(a_i - \theta_2^{\mathsf{T}} x_i\right)^2\right)\right| \\
&\quad = \left|\sum_{i=1}^{m} \left(a_i^2 - 2(\theta_1^{\mathsf{T}} x_i)a_i + (\theta_1^{\mathsf{T}} x_i)^2 - a_i^2 + 2(\theta_2^{\mathsf{T}} x_i)a_i - (\theta_2^{\mathsf{T}} x_i)^2\right)\right| \\
&\quad = \left|\sum_{i=1}^{m} \left(2a_i(\theta_2 - \theta_1)^{\mathsf{T}} x_i + (\theta_1^{\mathsf{T}} x_i)^2 - (\theta_2^{\mathsf{T}} x_i)^2\right)\right| \\
&\quad \leq \sum_{i=1}^{m} |2a_i(\theta_2 - \theta_1)^{\mathsf{T}} x_i| + \sum_{i=1}^{m} \left|(\theta_1^{\mathsf{T}} x_i)^2 - (\theta_2^{\mathsf{T}} x_i)^2\right| \\
&\quad \leq \sum_{i=1}^{m} 2|a_i| \cdot \|\theta_2 - \theta_1\| \cdot \|x_i\| + \sum_{i=1}^{m} \left|(\theta_1^{\mathsf{T}} x_i)^2 - (\theta_1^{\mathsf{T}} x_i)(\theta_2^{\mathsf{T}} x_i) + (\theta_2^{\mathsf{T}} x_i)(\theta_1^{\mathsf{T}} x_i) - (\theta_2^{\mathsf{T}} x_i)^2\right| \\
&\quad \leq 2m\|\theta_2 - \theta_1\| + \sum_{i=1}^{m} \left(|\theta_1^{\mathsf{T}} x_i| \cdot \|\theta_1 - \theta_2\| \cdot |x_i| + |\theta_2^{\mathsf{T}} x_i| \cdot \|\theta_1 - \theta_2\| \cdot |x_i|\right) \\
&\quad \leq 2m\|\theta_2 - \theta_1\| + 2m\|\theta_2 - \theta_1\| \\
&\quad \leq 4m\|\theta_2 - \theta_1\|.
\end{aligned}$$

□

The following technical lemma will be used in the proof.

Lemma 4 *Let m be a positive integer, and a and b be positive real numbers.*

1. *If $m > 2a \ln a$, then $m > a \ln m$; and*
2. *If $m \leq a \ln m + b$, then $m \leq 2a \ln a + 2b$.*

Proof The first part is proved by Fong (1995b). For the second part, the inequality $m \leq a \ln m + b$ can be written as $m' \leq a' \ln m'$, where $m' = me^{b/a}$ and $a' = ae^{b/a}$. Using the contrapositive of the first part, we have $m' \leq 2a' \ln a'$, which finishes the proof immediately. □

We are now ready to prove the theorem.

Proof of Theorem 1 For any real number $\tau \in [0, 1]$ let Θ_τ denote the set of $O((\sqrt{n}/\tau)^n)$ representatives $\theta \in [0, 1]^n$ chosen by uniform discretization so that for any $\theta' \in [0, 1]^n$, there exists a representative $\theta \in \Theta_\tau$ satisfying $\|\theta - \theta'\| \leq \tau$.

Our proof has three steps. The first is to bound the maximum number m of times the algorithm makes a prediction of $\perp$ in terms of the input parameters α_1, α_2, and the discretization rate τ. The second is to choose the parameters α_1, α_2, and τ. The third is to show that with high probability every prediction made by the algorithm that is not $\perp$ is accurate.

Step 1 Let X_t denote the matrix X during the tth timestep of the algorithm, similarly for z_t, $\bar{q}_t$, and $\bar{u}_t$. Let m denote the number of times that the algorithm makes a prediction of $\perp$. Let $\mathcal{T}$ denote the set of timesteps t during which a prediction of $\perp$ is made. From Lemma 13 of Auer (2002)[12] we have that $\sum_{t\in\mathcal{T}} \|\bar{q}_t\| \leq 2\sqrt{5nm \ln(m)}$ and $\sum_{t\in\mathcal{T}} \|\bar{u}_t\| \leq 5\sqrt{nm}$. However, by the operation of our algorithm, we also have that $\sum_{t\in\mathcal{T}} \|\bar{q}_t\| \geq m_q\alpha_1$ and $\sum_{t\in\mathcal{T}} \|\bar{u}_t\| \geq m_u\alpha_2$, where m_q and m_u are the number of timesteps during which the conditions $\|\bar{q}\| \leq \alpha_1$ and $\|\bar{u}\| \leq \alpha_2$ are violated, respectively. Hence, we have

$$m_q\alpha_1 \leq 2\sqrt{5nm \ln m}, \tag{6}$$

and

$$m_u\alpha_2 \leq 5\sqrt{nm}. \tag{7}$$

Furthermore, it is clear that $m_q + m_u \geq m$. Combining this inequality with (6) and (7), we get

$$m \leq \frac{2\sqrt{5nm \ln m}}{\alpha_1} + \frac{5\sqrt{nm}}{\alpha_2},$$

or, equivalently,

$$\sqrt{m} \leq \frac{2\sqrt{5n \ln m}}{\alpha_1} + \frac{5\sqrt{n}}{\alpha_2}.$$

Next, applying the elementary inequality $x + y \leq \sqrt{2x^2 + 2y^2}$ for any reals x, y, we have

$$m \leq \frac{40n \ln m}{\alpha_1^2} + \frac{50n}{\alpha_2^2}.$$

[12]The quantities $\bar{q}_t$, $\bar{u}_t$, n, $\mathcal{T}$, and m of our notation denote the same quantities, respectively, as $a_{i(t)}(t)$, $\tilde{v}_{i(t)}(t)$, d, $\Psi(T + 1)$, and $|\Psi(T + 1)|$ using the notation in the paper by Auer (2002).

Finally, we use Lemma 4 and obtain an upper bound $\bar{m}$ for m, where

$$\bar{m} = O\left(\frac{n\ln(n/\alpha_1)}{\alpha_1^2} + \frac{n}{\alpha_2^2}\right). \tag{8}$$

In fact, as we will specify later, $\alpha_1 = o(\epsilon^2/\sqrt{n}) \ll \alpha_2 = O(\epsilon)$, so we may safely let $\bar{m}$ be

$$\bar{m} = \frac{An\ln(n/\alpha_1)}{\alpha_1^2} \tag{9}$$

for some appropriate constant A.[13]

Step 2 We now set values for τ, α_1, and α_2. For the discretization rate τ, we choose

$$\tau = \frac{c_1\epsilon}{4\bar{m}} = O\left(\frac{\epsilon}{\bar{m}}\right),$$

where c_1 is a positive constant whose value we will specify later. We then choose $\alpha_2 = c_3\epsilon$ for some constant c_3 whose value is to be specified.

Finding an appropriate value for α_1 is trickier. We claim that, if we set

$$\alpha_1 = O\left(\frac{\epsilon^2}{\sqrt{n\ln\frac{1}{\delta'}\ln\frac{n\ln(1/\delta')}{\epsilon}}}\right) \tag{10}$$

where

$$\delta' := \left(\frac{\delta}{2\bar{m}}\right)\left(\frac{\tau}{\sqrt{n}}\right)^n \propto \left(\frac{\delta}{\bar{m}}\right)\left(\frac{\epsilon}{\bar{m}\sqrt{n}}\right)^n = O\left(\frac{\delta\epsilon^n}{(\bar{m}\sqrt{n})^n}\right), \tag{11}$$

then, for some constant $c_4 \in (0,1)$ whose value will be specified later, we have

$$\frac{\epsilon^2}{4\alpha_1^2} \geq 2\sqrt{8\bar{m}\ln(2/\delta')} + c_4\epsilon. \tag{12}$$

To see this, note that a sufficient condition for (12) to hold is

$$\frac{\epsilon^2}{\alpha_1} \geq \sqrt{B\ln\frac{1}{\delta'}n\ln\frac{n^2}{\alpha_1^2}}$$

for some constant B, where we have used (9) and the fact that $\sqrt{\bar{m}} \gg \epsilon$ (as a consequence of our choice that $\alpha_1 \ll \epsilon$). By elementary algebra, the above inequality can be rewritten as

$$\frac{n^2}{\alpha_1^2} \geq \frac{Bn^3}{\epsilon^4}\ln\frac{1}{\delta'}\ln\frac{n^2}{\alpha_1^2}. \tag{13}$$

[13] It is worth noting that it is *not* circular reasoning to make use of the relation between α_1 and α_2 before specifying their values. Here, our proof is a *constructive* one; namely, we aim to find values for α_1 and α_2 so that (8) is satisfied. Therefore, our "forward" use of the relation between α_1 and α_2 is correct as long as their specified values (in Step 2 of the proof) are consistent.

At this point, we may apply the first part of Lemma 4: using notation from this lemma, if we choose $m = \frac{n^2}{\alpha_1^2}$ and $a = \frac{Bn^3}{\epsilon^4} \ln \frac{1}{\delta'}$, then (13) holds as long as $m > 2a \ln a$, which can be converted into the form given by (10).

Step 3 Consider some fixed timestep t during the execution of Algorithm 1 such that the algorithm makes a valid prediction (*i.e.*, $\hat{y}_t \neq \perp$). Let $\hat{\theta}$ denote the solution of the norm-constrained least-squares minimization (line 6 in Algorithm 1). By definition, since $\perp$ was not predicted, we have that $\|\bar{q}_t\| \leq \alpha_1$ and $\|\bar{u}_t\| \leq \alpha_2$. We would like to show that $|\hat{\theta}^\mathsf{T} x_t - \theta^\mathsf{T} x_t| \leq \epsilon$ so that the accuracy requirement of Definition 1 is satisfied. Suppose not, namely that $|(\hat{\theta} - \theta)^\mathsf{T} x_t| > \epsilon$. Using Lemma 2, we can lower bound the quantity $\Delta_E(\hat{\theta})^2 = \sum_{i=1}^m [(\theta - \hat{\theta})^\mathsf{T} X(i)]^2$, where m denotes the number of rows of the matrix X. Applying Lemma 2, we have that $\epsilon \leq \alpha_1 \Delta_E(\hat{\theta}) + 2\alpha_2$, which by our choice of $\alpha_2 = c_3\epsilon$ is equivalent to

$$\Delta_E(\hat{\theta})^2 \geq \left(\frac{(1 - 2c_3)\epsilon}{\alpha_1} \right)^2. \tag{14}$$

Suppose we choose $c_3 = 1/4$. Then, by combining (12) with (14) we have that the following is satisfied (recall that $m \leq \bar{m}$ always holds)

$$\Delta_E(\hat{\theta})^2 \geq 2\sqrt{8m \ln(2/\delta')} + c_4\epsilon.$$

We are almost finished. Let $\mathrm{Rep}(\hat{\theta})$ denote the representative in Θ_τ that is closest (in ℓ_2 distance) to $\hat{\theta}$. By Lemma 3 applied with our choice of τ, we have that

$$\Delta_E(\mathrm{Rep}(\hat{\theta}))^2 \geq \Delta_E(\hat{\theta})^2 - c_1\epsilon.$$

Combining the two previous equations we have that $\Delta_E(\mathrm{Rep}(\hat{\theta}))^2 \geq 2\sqrt{8m \ln(2/\delta')} + (c_4 - c_1)\epsilon$. By Lemma 1 we have that

$$\sum_{i=1}^m \big(z_i - \mathrm{Rep}(\hat{\theta})^\mathsf{T} X(i)\big)^2 > \sum_{i=1}^m \big(z_i - \theta^\mathsf{T} X(i)\big)^2 + (c_4 - c_1)\epsilon$$

holds with probability at least $1 - 2\delta'$. With one more application of Lemma 3, this last equation implies that

$$\sum_{i=1}^m \left(z_i - \hat{\theta}^\mathsf{T} X(i)\right)^2 > \sum_{i=1}^m \big(z_i - \theta^\mathsf{T} X(i)\big)^2 + (c_4 - 2c_1)\epsilon. \tag{15}$$

Using $c_4 = 1$ and $c_1 = 1/4$, (15) becomes

$$\sum_{i=1}^m \big(z_i - \hat{\theta}^\mathsf{T} X(i)\big)^2 > \sum_{i=1}^m \big(z_i - \theta^\mathsf{T} X(i)\big)^2 + \frac{\epsilon}{2},$$

which contradicts the fact that $\hat{\theta}$ was chosen to minimize the term $\sum_{i=1}^m (z_i - \hat{\theta}^\mathsf{T} X(i))^2$. Our application of Lemma 1 is the only result we used that holds only with high probability. To ensure that it holds for the entire (infinite) execution of the algorithm we apply the union bound. Note that Lemma 1 is applied only to representative vectors in Θ_τ. There are $O((\sqrt{n}/\tau)^n)$ such vectors. Also note that since $m \leq \bar{m}$, there are at most $\bar{m}$ distinct pairs

(X_t, z_t) of samples used for training $\hat{\theta}$ during the entire execution of the algorithm. Hence, our proof involves the application of Lemma 1 at most $O(m(\sqrt{n}/\tau)^n)$ times. Our choice of δ' in (11) thus guarantees the total failure probability of the algorithm is at most δ. □

Appendix B: Proof of Theorem 2

Without loss of generality, assume $h^* \in \mathcal{H}_1$, and so $|\hat{y}_{t1} - y_t| \leq \epsilon/8$ whenever $\hat{y}_{t1} \neq \perp$. The following lemma says the accuracy requirement of Definition 1 is satisfied.

Lemma 5 *If $\hat{y}_t \neq \perp$, then $|\hat{y}_t - y_t| < \epsilon$.*

Proof By assumption, $|\hat{y}_{t1} - y_t| \leq \epsilon/8$. Since the midpoint prediction $\hat{y}_t$ differs from $\hat{y}_{t1}$ by at most $\epsilon/2$, the prediction error can be bounded using the triangle inequality $|a + b| \leq |a| + |b|$:

$$|\hat{y}_t - y_t| \leq |\hat{y}_t - \hat{y}_{t1}| + |\hat{y}_{t1} - y_t| \leq \frac{\epsilon}{2} + \frac{\epsilon}{8} < \epsilon. \qquad \square$$

We next show that the sample complexity requirement of Definition 1 is also satisfied. Note that the total number of $\perp$s returned by noisy union is the number of timesteps lines 11 and 17 are executed. Since line 11 can be executed for at most $\sum_i B(\epsilon/8, \delta/(k+1))$ times, all that remains is to show that line 17 cannot be executed many times. To do this, we first prove the following three lemmas.

Lemma 6 *Whenever line 17 of Algorithm 2 is executed, c_{1i} and Δ_{1i} will be updated for at least one i in $\{2, 3, \ldots, k\}$. Consequently, line 17 is executed for at most $m(k-1)$ timesteps.*

Proof Suppose at timestep t `noisy union` predicts $\hat{y}_t = \perp$ because $|\hat{y}_{ti} - \hat{y}_{tj}| \geq \epsilon$ for some $1 \leq i < j \leq k$. Then by the triangle inequality $|a| + |b| \geq |a - b|$, we have that $|\hat{y}_{t1} - \hat{y}_{ti}| + |\hat{y}_{t1} - \hat{y}_{tj}| \geq |\hat{y}_{ti} - \hat{y}_{tj}| \geq \epsilon$, which implies at least one of $|\hat{y}_{t1} - \hat{y}_{ti}| \geq \epsilon/2$ or $|\hat{y}_{t1} - \hat{y}_{tj}| \geq \epsilon/2$ is true. Hence, either c_{1i} and Δ_{1i}, or c_{1j} and Δ_{1j}, or both, are updated. □

We next turn to decide an appropriate value of m to guarantee that the correct hypothesis is not ruled out with high probability.

Lemma 7 *Let i be the index given in the Lemma 6 such that $|\hat{y}_{t1} - \hat{y}_{ti}| \geq \epsilon/2$. On average, Δ_{1i} is decremented by at least $\epsilon^2/8$.*

Proof By definition, the expected *increment* of Δ_{1i} is:

$$\begin{aligned}
&\mathbf{E}_{z_t \sim y_t}\left[(\hat{y}_{t1} - z_t)^2 - (\hat{y}_{ti} - z_t)^2\right] \\
&\quad = \mathbf{E}_{z_t \sim y_t}\left[(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} + \hat{y}_{ti} - 2z_t)\right] \\
&\quad = y_t(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} + \hat{y}_{ti} - 2) + (1 - y_t)(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} + \hat{y}_{ti}) \\
&\quad = -(\hat{y}_{t1} - \hat{y}_{ti})^2 + 2(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} - y_t) \\
&\quad \leq -(\hat{y}_{t1} - \hat{y}_{ti})^2 + 2|(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} - y_t)| \\
&\quad = -(\hat{y}_{t1} - \hat{y}_{ti})^2 + 2|\hat{y}_{t1} - \hat{y}_{ti}||\hat{y}_{t1} - y_t|
\end{aligned}$$

$$\leq -(\hat{y}_{t1} - \hat{y}_{ti})^2 + \frac{\epsilon}{4}|\hat{y}_{t1} - \hat{y}_{ti}|$$
$$= |\hat{y}_{t1} - \hat{y}_{ti}|\left(-|\hat{y}_{t1} - \hat{y}_{ti}| + \frac{\epsilon}{4}\right)$$
$$\leq \frac{\epsilon}{2}\left(-\frac{\epsilon}{2} + \frac{\epsilon}{4}\right) = -\frac{\epsilon^2}{8}.$$

□

Lemma 8 *Each change of* Δ_{1i} *in line* 20 *is at most* $2|\hat{y}_{t1} - \hat{y}_{ti}|$.

Proof From simple algebra, we have

$$\left|(\hat{y}_{t1} - z_t)^2 - (\hat{y}_{ti} - z_t)^2\right| = |(\hat{y}_{t1} - \hat{y}_{ti})(\hat{y}_{t1} + \hat{y}_{ti} - 2z_t)| = |\hat{y}_{t1} - \hat{y}_{ti}||\hat{y}_{t1} + \hat{y}_{ti} - 2z_t|.$$

It can be easily verified that $|\hat{y}_{t1} + \hat{y}_{ti} - 2z_t| < 2$ for $z_t \in \{0, 1\}$, and the lemma follows. □

Based on Lemmas 7 and 8, we can decide the value of m. To simplify notation, assume without loss of generality that $|\hat{y}_{t1} - \hat{y}_{ti}| \geq \epsilon/2$ for the first m timesteps; namely, Δ_{1i} changes in every timestep until one of $\mathbf{A}_1$ and $\mathbf{A}_i$ is eliminated (line 22 in Algorithm 2) at the end of timestep m. Using Hoeffding (1963)'s inequality applied to the martingale Δ_{1i}, we have at the end of timestep m that

$$\Pr(\Delta_{1i} \geq 0) \leq \Pr\left(\Delta_{1i} - \mathbf{E}[\Delta_{1i}] \geq \sum_{t=1}^{m} \frac{\epsilon_t^2}{2}\right)$$
$$\leq \exp\left(-\frac{(\sum_{t=1}^{m} \epsilon_t^2/2)^2}{2\sum_{t=1}^{\mathsf{T}}(2\epsilon_t)^2}\right)$$
$$= \exp\left(-\frac{\sum_{t=1}^{m} \epsilon_t^2}{32}\right)$$
$$\leq \exp\left(-\frac{m\epsilon^2}{128}\right)$$

where the first inequality is due to Lemma 7, the second due to Hoeffding's inequality and Lemma 8, and the last due to the fact that $\epsilon_t \geq \epsilon/2$. Setting the last expression to be at most δ/k^2, we can solve for m:

$$m = \left\lceil \frac{128}{\epsilon^2} \ln \frac{k^2}{\delta} \right\rceil.$$

By the union bound, we have that

$$\Pr(\Delta_{1i} > 0 \text{ for any } i \in \{2, 3, \ldots, k\}) \leq (k-1)\frac{\delta}{k^2} < \frac{\delta}{k+1}.$$

That is, the probability that the `noisy union` algorithm ends up with an incorrect hypothesis is tiny if every subalgorithm succeeds. Applying the union bound again with the k subalgorithms, $\mathbf{A}_i$, each failing with probability at most $\delta/(k+1)$, we conclude that the failure probability of `noisy union` is at most δ.

Using the above value for m and Lemma 6, the KWIK bound of `noisy union` is

$$m(k-1) + \sum_{i=1}^{m} B_i\left(\frac{\epsilon}{8}, \frac{\delta}{k+1}\right) = (k-1)\left\lceil \frac{128}{\epsilon^2} \ln \frac{k^2}{\delta} \right\rceil + \sum_{i=1}^{m} B_i\left(\frac{\epsilon}{8}, \frac{\delta}{k+1}\right).$$

Appendix C: Proof of Theorem 3

The proof we provide in this section relies on Proposition 1 of Strehl et al. (2006a). For convenience, we restate that proposition and two supporting definitions that are slightly modified to match our presentation.

Definition 3 (Strehl et al. 2006a) Let $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ be an MDP with a given set of action values, $Q(s,a)$, for each state–action (s,a), and a set K of state–actions, called the *known state–actions*. We define the *known state–action MDP* $M_K = \langle \mathcal{S}, \mathcal{A}, T_K, R_K, \gamma \rangle$ as follows: for all $(s,a) \in K$, $R_K(s,a) = R(s,a)$ and $T_K(\cdot|s,a) = T(\cdot|s,a)$; for all $(s,a) \notin K$, $R_K(s,a) = Q(s,a)(1-\gamma)$ and $T_K(s|s,a) = 1$.

The MDP $\hat{M}$ used in Algorithm 3 is similar to the known state–action MDP defined above. In fact, if we define the set K of known state–actions to be the set of state–actions in which both the transition function learner $\mathbf{A}_T$ and reward function learner $\mathbf{A}_R$ make non-$\perp$ predictions, then $\hat{M}$ is almost identical to M_K, except that the dynamics in $\hat{M}$ are estimated from experience. For this reason, $\hat{M}$ is sometimes called the *empirical known state–action MDP*.

Definition 4 (Strehl et al. 2006a) For algorithm $\mathbf{A}$, for each timestep t, let K_t (we drop the subscript t if t is clear from context) be a set of state–actions defined arbitrarily in a way that depends only on the history of the agent up to timestep t (before the t-th action). We define A_K to be the event, called the *escape event*, that some state–action $(s,a) \notin K_t$ is experienced by the agent at time t.

Theorem 4 (Strehl et al. 2006a) *Let $\mathbf{A}(\epsilon, \delta)$ be an algorithm that acts greedily according to $Q(\cdot,\cdot)$, whose value is denoted $Q_t(\cdot,\cdot)$ at timestep t. Define $V_t(s) = \max_a Q_t(s,a)$. Suppose that on every timestep t, there exists a set K_t of state–actions that depends only on the agent's history up to timestep t. We assume that $K_t = K_{t+1}$ unless, during timestep t, an update to some state–action value occurs or the escape event A_K happens. Let M_{K_t} be the known state–action MDP and π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \operatorname{argmax}_a Q_t(s,a)$. Suppose that for any inputs ϵ and δ, with probability at least $1-\delta$, the following conditions hold for all states s, actions a, and timesteps t: (1) $V_t(s) \geq V^*(s) - \epsilon$ (optimism), (2) $V_t(s) - V^{\pi_t}_{M_{K_t}}(s) \leq \epsilon$ (accuracy), and (3) the total number of updates of action–value estimates plus the number of times the escape event from K_t, A_K, can occur is bounded by $\zeta(\epsilon, \delta)$ (learning complexity). Then, when $\mathbf{A}(\epsilon, \delta)$ is executed on any MDP M, it will follow a 4ϵ-optimal policy from its current state on all but*

$$O\left(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$$

timesteps, with probability at least $1-2\delta$.

We first provide a version of the simulation lemma (Lemma 9), an inequality of Singh and Yee (1994) (Lemma 10), and then verify the three conditions in Theorem 4 in Lemmas 11–13 to show `KWIK-Rmax` is PAC-MDP.

Lemma 9 (Simulation Lemma) *Let $M_1 = \langle \mathcal{S}, \mathcal{A}, T_1, R_1, \gamma \rangle$ and $M_2 = \langle \mathcal{S}, \mathcal{A}, T_2, R_2, \gamma \rangle$ be two MDPs with the same state/action spaces and discount factor. Let Q_1^* and Q_2^* (V_1^* and V_2^*) be their optimal state–action value (state-value) functions, respectively. Assume the two transition functions and reward functions are close in the following sense: for every (s, a),*

$$|T_1(\cdot \mid s, a) - T_2(\cdot \mid s, a)| \leq \epsilon_T,$$
$$|R_1(s, a) - R_2(s, a)| \leq \epsilon_R,$$

where $|T_1(\cdot \mid s, a) - T_2(\cdot \mid s, a)|$ is as defined in Definition 2, then for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$:

$$|Q_1^*(s, a) - Q_2^*(s, a)| \leq \frac{\epsilon_R(1-\gamma) + \epsilon_T}{(1-\gamma)^2},$$
$$|V_1^*(s) - V_2^*(s)| \leq \frac{\epsilon_R(1-\gamma) + \epsilon_T}{(1-\gamma)^2}.$$

Proof We will prove the case where $\mathcal{S}$ is uncountable; the other case is similar. Define the *Bellman operators*, $\mathcal{B}_1$ and $\mathcal{B}_2$, for M_1 and M_2, respectively: for $i = 1, 2$ and any state–action value function $Q : \mathcal{S} \times \mathcal{A} \to \Re$,

$$\mathcal{B}_i Q(s, a) = R_i(s, a) + \gamma \int_{s' \in \mathcal{S}} T(s' \mid s, a) \sup_{a' \in \mathcal{A}} Q(s', a') ds'.$$

It is known that Q_i^* is the *fixed point* of $\mathcal{B}_i$: $\mathcal{B}_i Q_i^* = Q_i^*$. Define two errors: the ℓ_∞ approximation error $e = \|Q_1^* - Q_2^*\|_\infty$ and the ℓ_∞ Bellman backup error $b = \|\mathcal{B}_1 Q_2^* - \mathcal{B}_2 Q_2^*\|_\infty$. Then,

$$\begin{aligned} e &= \|\mathcal{B}_1 Q_1^* - \mathcal{B}_2 Q_2^*\|_\infty \\ &\leq \|\mathcal{B}_1 Q_1^* - \mathcal{B}_1 Q_2^*\|_\infty + \|\mathcal{B}_1 Q_2^* - \mathcal{B}_2 Q_2^*\|_\infty \\ &\leq \gamma \|Q_1^* - Q_2^*\|_\infty + \|\mathcal{B}_1 Q_2^* - \mathcal{B}_2 Q_2^*\|_\infty \\ &= \gamma e + b, \end{aligned}$$

where the first step is due to the fixed-point property of $\mathcal{B}_i$, the second due to the triangle inequality, the third due to the contraction property of $\mathcal{B}_i$ in the ℓ_∞ norm (Puterman 1994), and the last due to the definitions of e and b. It follows immediately that $(1-\gamma)e \leq b$, which leads to

$$e \leq \frac{b}{1-\gamma}. \tag{16}$$

We now give an upper bound for b:

$$\begin{aligned} b &= \sup_{s,a} |\mathcal{B}_1 Q_2^*(s, a) - \mathcal{B}_2 Q_2^*(s, a)| \\ &= \sup_{s,a} \left| \big(R_1(s, a) - R_2(s, a)\big) + \gamma \int_{\mathcal{S}} \big(T_1(s' \mid s, a) - T_2(s' \mid s, a)\big) \sup_{a'} Q_2^*(s', a') ds' \right| \end{aligned}$$

$$
\begin{aligned}
&\leq \sup_{s,a} |R_1(s,a) - R_2(s,a)| + \gamma \sup_{s,a} \left| \int_{\mathcal{S}} \big(T_1(s' \mid s,a) - T_2(s' \mid s,a)\big) \sup_{a'} Q_2^*(s',a') ds' \right| \\
&\leq \epsilon_R + \gamma \sup_{s,a} \int_{\mathcal{S}} \big|T_1(s' \mid s,a) - T_2(s' \mid s,a)\big| \sup_{a'} |Q_2^*(s',a')| ds' \\
&\leq \epsilon_R + \frac{\gamma}{1-\gamma} \sup_{s,a} \int_{\mathcal{S}} \big|T_1(s' \mid s,a) - T_2(s' \mid s,a)\big| ds' \\
&\leq \epsilon_R + \frac{\gamma}{1-\gamma} \epsilon_T \\
&\leq \frac{\epsilon_R(1-\gamma) + \epsilon_T}{1-\gamma},
\end{aligned}
$$

where the first inequality is due to the triangle inequality and the fact that $\sup_x\{f_1(x) + f_2(x)\} \leq \sup_x f_1(x) + \sup_x f_2(x)$ for all real-valued functions f_1 and f_2, the second due to the Cauchy-Schwartz inequality, the third due to $\|Q_2^*\|_\infty \leq \gamma/(1-\gamma)$ since the reward function is nonnegative and upper-bounded by 1 and has discount factor γ. Combining this result with (16), we have for all (s,a) that

$$
|Q_1^*(s,a) - Q_2^*(s,a)| \leq e \leq \frac{b}{1-\gamma} \leq \frac{\epsilon_R(1-\gamma) + \epsilon_T}{(1-\gamma)^2}.
$$

The second part of the lemma follows immediately from the following relation between optimal state–action value functions and optimal state-value functions: for any $s \in \mathcal{S}$,

$$
|V_1^*(s) - V_2^*(s)| = \left| \sup_a Q_1^*(s,a) - \sup_a Q_2^*(s,a) \right| \leq \sup_a |Q_1^*(s,a) - Q_2^*(s,a)|. \qquad \square
$$

The following lemma, which will be useful in our proof, is due to Singh and Yee (1994).[14] Although they consider finite MDPs only, their proof is also valid for arbitrary MDPs as long as the value functions and state transition probabilities used in their proof are all well-defined when the MDP under consideration has a continuous state space.

Lemma 10 (Singh and Yee 1994, Corollary 2) *Let $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ be an MDP, Q a value function, and π_Q the greedy policy with respect to Q. The true value functions of policy π_Q are denoted V^{π_Q} and Q^{π_Q}. If $|Q^*(s,a) - Q(s,a)| \leq \epsilon$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, then for all $s \in \mathcal{S}$,*

$$
V^*(s) - V^{\pi_Q}(s) = Q^*(s, \pi^*(s)) - Q^{\pi_Q}(x, \pi_Q(s)) \leq \frac{2\epsilon}{1-\gamma}.
$$

Lemma 11 *With probability at least $1 - \delta/2$, $Q(s,a) \geq Q^*(s,a) - \epsilon/4$ for all t and (s,a).*

Proof Since `KWIK-Rmax` (Algorithm 3) computes an ϵ_P-accurate Q_t function, we have

$$
Q_t(s,a) - Q^*(s,a) \geq Q_{\hat{M}}^*(s,a) - \epsilon_P - Q^*(s,a).
$$

[14] Note there is a typo in the definition of the loss function $L_{\tilde{Q}}$ on age 231. Using their notation, the definition should read: $L_{\tilde{Q}}(x) := Q^*(x, \pi^*(x)) - Q_{\pi_{\tilde{Q}}}(x, \pi_{\tilde{Q}}(x))$, where x is a state, $\pi_{\tilde{Q}}$ is the greedy policy with respect to $\tilde{Q}$.

We next bound $Q^*_{\hat{M}}(s,a) - Q^*(s,a)$. Let M_K be the known state–action MDP whose transition and reward functions agree with $\hat{M}$ in unknown state–actions and with M in known state–actions. Since the transition and reward functions of $\hat{M}$ are accurate (with failure probability at most $\delta_R + \delta_T = \delta/2$), Lemma 9 yields

$$\left|Q^*_{\hat{M}}(s,a) - Q^*_{M_K}(s,a)\right| \leq \frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2}.$$

On the other hand, since M_K is identical to M except in unknown state–actions where the largest possible rewards are assigned (line 5), its optimal state–action value function must be optimistic: $Q^*_{M_K}(s,a) \geq Q^*(s,a)$ for all (s,a). Combining these inequalities, we have

$$\begin{aligned} Q_t(s,a) - Q^*(s,a) &\geq Q^*_{\hat{M}}(s,a) - \epsilon_P - Q^*(s,a) \\ &\geq Q^*_{M_K}(s,a) - \frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2} - \epsilon_P - Q^*(s,a) \\ &\geq -\frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2} - \epsilon_P. \end{aligned}$$

The lemma follows by using the values of ϵ_R, ϵ_T, and ϵ_P given in the theorem. □

Lemma 12 *With probability at least* $1-\delta/2$, $V_t(s_t) - V^{\pi_t}_{M_K}(s_t) \leq \epsilon/4$, *where* $V_t(s) = \max_a Q_t(s,a)$, *and* π_t *is the policy computed by* `KWIK-Rmax` *at timestep* t.

Proof If Q_t is ϵ_P-accurate, then V_t is also ϵ_P-accurate:

$$\left|V_t(s) - V^*_{\hat{M}}(s)\right| = \left|\sup_a Q_t(s,a) - \sup_a Q^*_{\hat{M}}(s,a)\right| \leq \sup_a \left|Q_t(s,a) - Q^*_{\hat{M}}(s,a)\right| \leq \epsilon_P.$$

Consequently,

$$\begin{aligned} V_t(s) - V^{\pi_t}_{M_K}(s) &\leq V_t(s) - V^{\pi_t}_{\hat{M}}(s) + \frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2} \\ &\leq V^*_{\hat{M}}(s) + \epsilon_P - V^{\pi_t}_{\hat{M}}(s) + \frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2} \\ &\leq \frac{2\epsilon_P}{1-\gamma} + \epsilon_P + \frac{\epsilon_R(1-\gamma)+\epsilon_T}{(1-\gamma)^2} \;\leq\; \frac{\epsilon}{4}, \end{aligned}$$

where the first step is from Lemma 9, and the second from the ϵ_P-accuracy of V_t in MDP $\hat{M}$, and the third is due to Lemma 10. The only failure probability is in the use of Lemma 9, which is at most $\delta_T + \delta_R = \delta/2$. □

Lemma 13 *The total number of updates of* Q_t *plus the number of timesteps an unknown state is visited, denoted by* $\zeta(\epsilon,\delta)$, *is at most* $2(B_T(\epsilon_T,\delta_T) + B_R(\epsilon_R,\delta_R))$.

Proof Since Q_t (and also $\hat{M}$) is unchanged unless $\mathbf{A}_T$ or $\mathbf{A}_R$ acquire new samples, the number of timesteps Q_t changes is at most the total number of samples received by $\mathbf{A}_T$

and $\mathbf{A}_R$,[15] which is $B_T(\epsilon_T, \delta_T) + B_R(\epsilon_R, \delta_R)$. On the other hand, the number of timesteps an unknown state is visited is also upper bounded by $B_T(\epsilon_T, \delta_T) + B_R(\epsilon_R, \delta_R)$. The lemma then follows. □

We can now complete the proof. Using the previous lemmas and Proposition 1 from Strehl et al. (2006a), the sample complexity of exploration of `KWIK-Rmax` is

$$O\left(\frac{B_T(\epsilon(1-\gamma)^2, \delta) + B_R(\epsilon(1-\gamma), \delta)}{\epsilon(1-\gamma)^2} \ln\frac{1}{\delta} \ln\frac{1}{\epsilon(1-\gamma)}\right).$$

References

Abbeel, P., & Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the twenty-second international conference on machine learning* (pp. 1–8).

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Angluin, D. (2004). Queries revisited. *Theoretical Computer Science*, *313*, 175–194.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, *3*, 397–422.

Bagnell, J., Ng, A. Y., & Schneider, J. (2001). *Solving uncertain Markov decision problems* (Technical Report CMU-RI-TR-01-25). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Bellman, R. (1957). *Dynamic programming*. Princeton: Princeton University Press.

Bertsekas, D., & Shreve, S. (1978). *Stochastic optimal control: The discrete time case*. New York: Academic Press.

Blum, A. (1994). Separating distribution-free and mistake-bound learning models over the Boolean domain. *SIAM Journal on Computing*, *23*, 990–1000.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Brunskill, E., Leffler, B. R., Li, L., Littman, M. L., & Roy, N. (2008). CORL: A continuous-state offset-dynamics reinforcement learner. In *Proceedings of the twenty-fourth conference on uncertainty in artificial intelligence (UAI-08)* (pp. 53–61).

Brunskill, E., Leffler, B. R., Li, L., Littman, M. L., & Roy, N. (2009). Provably efficient learning with typed parametric models. *Journal of Machine Learning Research*, *10*, 1955–1988.

Cesa-Bianchi, N., Lugosi, G., & Stoltz, G. (2005). Minimizing regret with label efficient prediction. *IEEE Transactions on Information Theory*, *51*, 2152–2162.

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, *7*, 1205–1230.

Cesa-Bianchi, N., Gentile, C., & Orabona, F. (2009). Robust bounds for classification via selective sampling. In *Proceedings of the twenty-sixth international conference on machine learning (ICML-09)* (pp. 121–128).

Chow, C.-S., & Tsitsiklis, J. N. (1989). The complexity of dynamic programming. *Journal of Complexity*, *5*, 466–488.

Cohn, D. A., Atlas, L., & Ladner, R. E. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, *5*, 142–150.

[15] For randomized planners such as sparse sampling (Kearns et al. 2002), an additional parameter δ_P may be needed to deal with the failure probability in planning (line 10 of Algorithm 3, but our analysis still holds with the minor changes in δ_T and δ_R. For local planners such as sparse sampling (Kearns et al. 2002), we have made an implicit assumption that once the approximate value of a state is computed, the value is not re-computed the next time the state is visited, unless the empirical known state–action MDP $\hat{M}$ changes.

Diuk, C., Li, L., & Leffler, B. R. (2009). The adaptive k-meteorologists problem and its application to structure discovery and feature selection in reinforcement learning. In *Proceedings of the twenty-sixth international conference on machine learning (ICML-09)* (pp. 249–256).

Fong, P. W. L. (1995a). A quantitative study of hypothesis selection. In *Proceedings of the twelfth international conference on machine learning (ICML-95)* (pp. 226–234).

Fong, P. W. L. (1995b). *A quantitative study of hypothesis selection*. Master's thesis, Department of Computer Science, University of Waterloo, Ontario, Canada.

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997a). Using and combining predictors that specialize. In *STOC'97: Proceedings of the twenty-ninth annual ACM symposium on theory of computing* (pp. 334–343).

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997b). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Freund, Y., Mansour, Y., & Schapire, R. E. (2004). Generalization bounds for averaged classifiers. *The Annals of Statistics*, *32*, 1698–1722.

Golub, G. H., & Van Loan, C. F. (1989). *Matrix computations* (2nd ed.). Baltimore: The Johns Hopkins University Press.

Helmbold, D. P., Littlestone, N., & Long, P. M. (2000). Apple tasting. *Information and Computation*, *161*, 85–139.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, University College London.

Kakade, S., Kearns, M., & Langford, J. (2003). Exploration in metric state spaces. In *Proceedings of the 20th international conference on machine learning*.

Kearns, M. J., & Koller, D. (1999). Efficient reinforcement learning in factored MDPs. In *Proceedings of the 16th International joint conference on artificial intelligence (IJCAI)* (pp. 740–747).

Kearns, M. J., & Schapire, R. E. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, *48*, 464–497.

Kearns, M. J., & Singh, S. P. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Kearns, M. J., Schapire, R. E., & Sellie, L. (1994). Toward efficient agnostic learning. *Machine Learning*, *17*, 115–141.

Kearns, M. J., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, *49*, 193–208.

Klasner, N., & Simon, H. U. (1995). From noise-free to noise-tolerant and from on-line to batch learning. In *Proceedings of the eighth annual conference on computational learning theory (COLT-95)* (pp. 250–257).

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Proceedings of the seventeenth European conference on machine learning (ECML-06)* (pp. 282–293).

Lane, T., & Brodley, C. E. (2003). An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, *51*, 73–107.

Leffler, B. R., Littman, M. L., & Edmunds, T. (2007). Efficient reinforcement learning with relocatable action models. In *Proceedings of the twenty-second conference on artificial intelligence (AAAI-07)*.

Li, L. (2009). *A unifying framework for computational reinforcement learning theory*. Doctoral dissertation, Rutgers University, New Brunswick, NJ.

Li, L., & Littman, M. L. (2010). Reducing reinforcement learning to KWIK online regression. *Annals of Mathematics and Artificial Intelligence*. doi:10.1007/s10472-010-9201-2.

Li, L., Littman, M. L., & Walsh, T. J. (2008). Knows what it knows: A framework for self-aware learning. In *Proceedings of the twenty-fifth international conference on machine learning* (pp. 568–575).

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the nineteenth international conference on World Wide Web (WWW-10)* (pp. 661–670).

Littlestone, N. (1987). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N. (1989). From on-line to batch learning. In *Proceedings of the second annual workshop on computational learning theory (COLT-89)* (pp. 269–284).

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, *13*, 103–130.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York: Wiley.

Seung, H. S., Opper, M., & Tishby, N. (1992). Query by committee. In *Proceedings of the fifth annual workshop on computational learning theory (COLT-92)* (pp. 287–294).

Shafer, G., & Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, *9*, 371–421.

Singh, S. P., & Yee, R. C. (1994). An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, *16*, 227.

Sontag, E. D. (1998). *Texts in Applied Mathematics: Vol. 6. Mathematical control theory: Deterministic finite dimensional systems* (2nd ed.). Berlin: Springer.

Strehl, A. L., & Littman, M. L. (2008). Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems, 20*.

Strehl, A. L., Li, L., & Littman, M. L. (2006a). Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the 22nd conference on uncertainty in artificial intelligence (UAI 2006)*.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., & Littman, M. L. (2006b). PAC model-free reinforcement learning. In *Proceedings of the twenty-third international conference on machine learning (ICML-06)*.

Strehl, A. L., Mesterharm, C., Littman, M. L., & Hirsh, H. (2006c). Experience-efficient learning in associative bandit problems. In *Proceedings of the twenty-third international conference on machine learning (ICML-06)*.

Strehl, A. L., Diuk, C., & Littman, M. L. (2007). Efficient structure learning in factored-state MDPs. In *Proceedings of the twenty-second national conference on artificial intelligence (AAAI-07)*

Strehl, A. L., Li, L., & Littman, M. L. (2009). Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, *10*, 2413–2444.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: The MIT Press.

Szita, I., & Szepesvári, C. (2010). Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the twenty-seventh international conference on machine learning (ICML-2010)*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*, 1134–1142.

Walsh, T. J., Szita, I., Diuk, C., & Littman, M. L. (2009). Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence (UAI-09)* (pp. 591–598). A refined version is available as Technical Report DCS-tr-660, Department of Computer Science, Rutgers University, December, 2009.

Weiss, G. M., & Tian, Y. (2006). Maximizing classifier utility when training data is costly. *SIGKDD Explorations*, *8*, 31–38.