Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning

Tom Zahavy *
Dept. of Electrical Engineering
The Technion - Israel Institute of Technology
Israel, Haifa 3200003
tomzahavy@campus.technion.ac.il

Matan Haroush *
Dept. of Electrical Engineering
The Technion - Israel Institute of Technology
Israel, Haifa 3200003
matan.h@campus.technion.ac.il

Nadav Merlis *
Dept. of Electrical Engineering
The Technion - Israel Institute of Technology
Israel, Haifa 3200003
merlis@campus.technion.ac.il

Daniel J. Mankowitz
Deepmind
London, UK
dmankowitz@google.com

Shie Mannor
Dept. of Electrical Engineering
The Technion - Israel Institute of Technology
Israel, Haifa 3200003
shie@ee.technion.ac.il

*** Equal contibution.**

Abstract

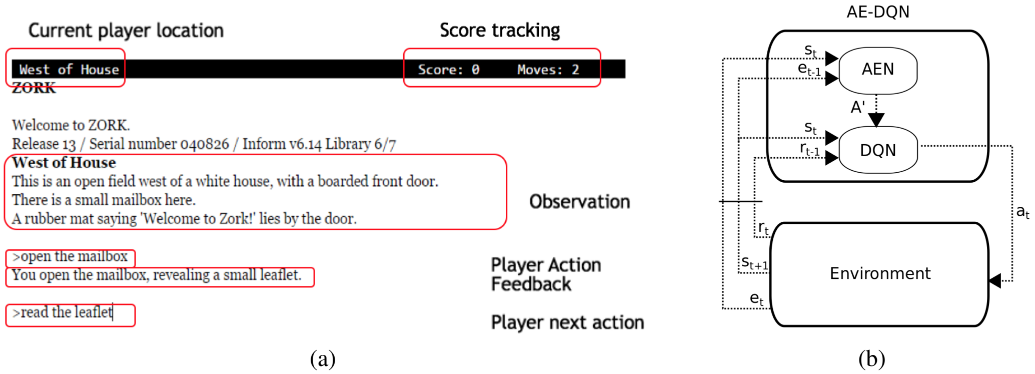
Learning how to act when there are many available actions in each state is a challenging task for Reinforcement Learning (RL) agents, especially when many of the actions are redundant or irrelevant. In such cases, it is sometimes easier to learn which actions **not** to take. In this work, we propose the Action-Elimination Deep Q-Network (AE-DQN) architecture that combines a Deep RL algorithm with an Action Elimination Network (AEN) that eliminates sub-optimal actions. The AEN is trained to predict invalid actions, supervised by an external elimination signal provided by the environment. Simulations demonstrate a considerable speedup and added robustness over vanilla DQN in text-based games with over a thousand discrete actions.

1 Introduction

Learning control policies for sequential decision-making tasks where both the state space and the action space are vast is critical when applying Reinforcement Learning (RL) to real-world problems. This is because there is an exponential growth of computational requirements as the problem size increases, known as the curse of dimensionality (Bertsekas and Tsitsiklis, 1995). Deep RL (DRL) tackles the curse of dimensionality due to large state spaces by utilizing a Deep Neural Network (DNN) to approximate the value function and/or the policy. This enables the agent to generalize across states without domain-specific knowledge (Tesauro, 1995; Mnih et al., 2015).

Despite the great success of DRL methods, deploying them in real-world applications is still limited. One of the main challenges towards that goal is dealing with large action spaces, especially when

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.



```
West of House                                   Score: 0      Moves: 2
ZORK

Welcome to ZORK.
Release 13 / Serial number 040826 / Inform v6.14 Library 6/7
West of House
This is an open field west of a white house, with a boarded front door.
There is a small mailbox here.
A rubber mat saying 'Welcome to Zork!' lies by the door.

>open the mailbox
You open the mailbox, revealing a small leaflet.

>read the leaflet
```

Figure 1: (a) **Zork interface.** The state in the game (observation) and the player actiosn are describe in natural language. (b) **The action elimination framework.** Upon taking action a_t, the agent observes a reward r_t, the next state s_{t+1} and an elimination signal e_t. The agent uses this information to learn two function approximation deep networks: a DQN and an AEN. The AEN provides an admissible actions set A' to the DQN, which uses this set to decide how to act and learn.

many of the actions are redundant or irrelevant (for many states). While humans can usually detect the subset of feasible actions in a given situation from the context, RL agents may attempt irrelevant actions or actions that are inferior, thus wasting computation time. Control systems for large industrial processes like power grids (Wen, O'Neill, and Maei, 2015; Glavic, Fonteneau, and Ernst, 2017; Dalal, Gilboa, and Mannor, 2016) and traffic control (Mannion, Duggan, and Howley, 2016; Van der Pol and Oliehoek, 2016) may have millions of possible actions that can be applied at every time step. Other domains utilize natural language to represent the actions. These action spaces are typically composed of all possible sequences of words from a fixed size dictionary resulting in considerably large action spaces. Common examples of systems that use this action space representation include conversational agents such as personal assistants (Dhingra et al., 2016; Li et al., 2017; Su et al., 2016; Lipton et al., 2016b; Liu et al., 2017; Zhao and Eskenazi, 2016; Wu et al., 2016), travel planners (Peng et al., 2017), restaurant/hotel bookers (Budzianowski et al., 2017), chat-bots (Serban et al., 2017; Li et al., 2016) and text-based game agents (Narasimhan, Kulkarni, and Barzilay, 2015; He et al., 2015; Zelinka, 2018; Yuan et al., 2018; Côté et al., 2018). RL is currently being applied in all of these domains, facing new challenges in function approximation and exploration due to the larger action space.

In this work, we propose a new approach for dealing with large actions spaces that is based on *action elimination*; that is, restricting the available actions in each state to a subset of the most likely ones (Figure 1(b)). We propose a method that eliminates actions by utilizing an auxiliary elimination signal which incorporates domain-specific prior knowledge regarding actions that can be eliminated. In many domains, creating an elimination signal can be done using rule-based systems. For example, in parser-based text games, the parser gives feedback regarding irrelevant actions *after* the action is played (e.g., Player: "Climb the tree." Parser: "There are no trees to climb"). Given such signal, we can train a machine learning model to predict it and then use it to generalize to unseen states. The core assumption in our approach is that it is easier to predict which actions are invalid or inferior in each state and leverage that information for control.

More specifically, we propose a system that learns an approximation of the Q-function and *concurrently learns to eliminate actions*. We focus on tasks where natural language characterizes both the states and the actions. In addition to language representation (for states and actions), the action space is combinatorial and irrelevant actions must be eliminated in order to learn. We introduce a novel DRL approach with two DNNs, a DQN and an Action Elimination Network (AEN), both designed using a Convolutional Neural Network (CNN) that is suited to NLP tasks (Kim, 2014). Using the last layer activations of the AEN, we design a linear contextual bandit model that eliminates irrelevant actions with high probability, balancing exploration/exploitation, and allowing the DQN to explore and learn Q-values only for valid actions.

We tested our method in a text-based game called "Zork". This game takes place in a virtual world in which the player interacts with the world through a text-based interface (see Figure 1(a)). The player can type in any command, corresponding to the in-game action. Since the input is text-based, this yields more than a thousand possible actions in each state (e.g., "open door", "open mailbox" etc.). We demonstrate the agent's ability to advance in the game faster than the baseline agents by eliminating irrelevant actions.

2 Related Work

Text-Based Games (TBG): Video games, via interactive learning environments like the Arcade Learning Environment (ALE) (Bellemare et al., 2013), have been fundamental to the development of DRL algorithms. Before the ubiquitousness of graphical displays, TBG like Zork were popular in the adventure gaming and role-playing communities. TBG present complex, interactive simulations which use simple language to describe the state of the environment, as well as reporting the effects of player actions (See Figure 1(a)). Players interact with the environment through text commands that respect a predefined grammar, which must be discovered in each game.

TBG provide a testbed for research at the intersection of RL and NLP, presenting a broad spectrum of challenges for learning algorithms (Côté et al., 2018)[1]. In addition to language understanding, successful play generally requires long-term memory, planning, exploration (Yuan et al., 2018), affordance extraction (Fulda et al., 2017), and common sense. Text games also highlight major open challenges for RL: the action space (text) is combinatorial and compositional, while game states are partially observable since text is often ambiguous or under-specific. Also, TBG often introduce stochastic dynamics, which is currently missing in standard benchmarks (Machado et al., 2017). For example, in Zork, there is a random probability of a troll killing the player. A thief can appear (also randomly) in each room.

Representations for TBG: To learn control policies from high-dimensional complex data such as text, good word representations are necessary. Kim (2014) designed a shallow word-level CNN and demonstrated state-of-the-art results on a variety of classification tasks by using word embeddings. For classification tasks with millions of labeled data, random embeddings were shown to outperform state-of-the-art techniques (Zahavy et al., 2018). On smaller data sets, using *word2vec* (Mikolov et al., 2013) yields good performance (Kim, 2014).

Previous work on TBG used pre-trained embeddings directly for control (Kostka et al., 2017; Fulda et al., 2017). Other works combined pre-trained embeddings with neural networks. For example, He et al. (2015) proposed to use Bag Of Words features as an input to a neural network, learned separate embeddings for states and actions, and then computed the Q function from autocorrelations between these embeddings. Narasimhan et al. (2015) suggested to use a word level Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) to learn a representation end-to-end, and Zelinka et al. (2018), combined these approaches.

DRL with linear function approximation: DRL methods such as the DQN have achieved state-of-the-art results in a variety of challenging, high-dimensional domains. This success is mainly attributed to the power of deep neural networks to learn rich domain representations for approximating the value function or policy (Mnih et al., 2015; Zahavy, Ben-Zrihem, and Mannor, 2016; Zrihem, Zahavy, and Mannor, 2016). Batch reinforcement learning methods with linear representations, on the other hand, are more stable and enjoy accurate uncertainty estimates. Yet, substantial feature engineering is necessary to achieve good results. A natural attempt at getting the best of both worlds is to learn a (linear) control policy on top of the representation of the last layer of a DNN. This approach was shown to refine the performance of DQNs (Levine et al., 2017) and improve exploration (Azizzadenesheli, Brunskill, and Anandkumar, 2018). Similarly, for contextual linear bandits, Riquelme et al. showed that a neuro-linear Thompson sampling approach outperformed deep (and linear) bandit algorithms in practice (Riquelme, Tucker, and Snoek, 2018).

RL in Large Action Spaces: Being able to reason in an environment with a large number of discrete actions is essential to bringing reinforcement learning to a larger class of problems. Most of the prior work concentrated on factorizing the action space into binary subspaces (Pazis and Parr, 2011; Dulac-Arnold et al., 2012; Lagoudakis and Parr, 2003). Other works proposed to embed the discrete

[1] See also The CIG Competition for General Text-Based Adventure Game-Playing Agents

actions into a continuous space, use a continuous-action policy gradient to find optimal actions in the continuous space, and finally, choose the nearest discrete action (Dulac-Arnold et al., 2015; Van Hasselt and Wiering, 2009). He et. al. (2015) extended DQN to unbounded (natural language) action spaces. His algorithm learns representations for the states and actions with two different DNNs and then models the Q values as an inner product between these representation vectors. While this approach can generalize to large action spaces, in practice, they only considered a small number of available actions (4) in each state.

Learning to eliminate actions was first mentioned by (Even-Dar, Mannor, and Mansour, 2003) who studied elimination in multi-armed bandits and tabular MDPs. They proposed to learn confidence intervals around the value function in each state and then use it to eliminate actions that are not optimal with high probability. Lipton et al. (2016a) studied a related problem where an agent wants to avoid catastrophic forgetting of dangerous states. They proposed to learn a classifier that detects hazardous states and then use it to shape the reward of a DQN agent. Fulda et al. (2017) studied affordances, the set of behaviors enabled by a situation, and presented a method for affordance extraction via inner products of pre-trained word embeddings.

3 Action Elimination

We now describe a learning algorithm for MDPs with an elimination signal. Our approach builds on the standard RL formulation (Sutton and Barto, 1998). At each time step t, the agent observes a state s_t and chooses a discrete action $a_t \in \{1, .., |A|\}$. After executing the action, the agent obtains a reward $r_t(s_t, a_t)$ and observes the next state s_{t+1} according to a transition kernel $P(s_{t+1}|s_t, a_t)$. The goal of the algorithm is to learn a policy $\pi(a|s)$ that maximizes the discounted cumulative return $V^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)|s_0 = s]$ where $0 < \gamma < 1$ is the discount factor and V is the value function. The optimal value function is given by $V^*(s) = \max_\pi V^\pi(s)$ and the optimal policy by $\pi^*(s) = \arg\max_\pi V^\pi(s)$. The Q-function $Q^\pi(s, a) = \mathbb{E}^\pi[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$ corresponds to the value of taking action a in state s and continuing according to policy π. The optimal Q-function $Q^*(s, a) = Q^{\pi^*}(s, a)$ can be found using the Q-learning algorithm (Watkins and Dayan, 1992), and the optimal policy is given by $\pi^*(s) = \arg\max_a Q^*(s, a)$.

After executing an action, the agent also observes a binary elimination signal $e(s, a)$, which equals 1 if action a may be eliminated in state s; that is, any optimal policy in state s will never choose action a (and 0 otherwise). The elimination signal can help the agent determine which actions not to take, thus aiding in mitigating the problem of large discrete action spaces. We start with the following definitions:

Definition 1. *Valid state-action pairs with respect to an elimination signal are state action pairs which the elimination process should not eliminate.*

As stated before, we assume that the set of valid state-action pairs contains all of the state-action pairs that are a part of some optimal policy, i.e., only strictly suboptimal state-actions can be invalid.

Definition 2. *Admissible state-action pairs with respect to an elimination algorithm are state action pairs which the elimination algorithm does not eliminate.*

In the following section, we present the main advantages of action elimination in MDPs with large action spaces. Afterward, we show that under the framework of linear contextual bandits (Chu et al., 2011), probability concentration results (Abbasi-Yadkori, Pal, and Szepesvari, 2011) can be adapted to guarantee that action elimination is correct in high probability. Finally, we prove that Q-learning coupled with action elimination will still converge.

3.1 Advantages in action elimination

Action elimination allows the agent to overcome some of the main difficulties in large action spaces, namely: Function Approximation and Sample Complexity.

Function Approximation: It is well known that errors in the Q-function estimates may cause the learning algorithm to converge to a suboptimal policy, a phenomenon that becomes more noticeable in environments with large action spaces (Thrun and Schwartz, 1993). Action elimination may mitigate this effect by taking the $\max$ operator only on valid actions, thus, reducing potential overestimation errors. Another advantage of action elimination is that the Q-estimates need only be accurate for valid actions. The gain is two-fold: first, there is no need to sample invalid actions for the function approximation to converge. Second, the function approximation can learn a simpler mapping (i.e., only the Q-values of the valid state-action pairs), and therefore may converge faster and to a better solution by ignoring errors from states that are not explored by the Q-learning policy (Hester et al., 2018).

Sample Complexity: The sample complexity of the MDP measures the number of steps, during learning, in which the policy is not ϵ-optimal (Kakade and others, 2003). Assume that there are A' actions that should be eliminated and are ϵ-optimal, i.e., their value is at least $V^*(s) - \epsilon$. According to lower bounds by (Lattimore and Hutter, 2012), We need at least $\epsilon^{-2}(1-\gamma)^{-3}\log 1/\delta$ samples per state-action pair to converge with probability $1-\delta$. If, for example, the eliminated action returns no reward and doesn't change the state, the action gap is $\epsilon = (1-\gamma)V^*(s)$, which translates to $V^*(s)^{-2}(1-\gamma)^{-5}\log 1/\delta$ 'wasted' samples for learning each invalid state-action pair. For large γ, this can lead to a tremendous number of samples (e.g., for $\gamma = 0.99, \ \ (1-\gamma)^{-5} = 10^{10}$). Practically, elimination algorithms can eliminate these actions substantially faster, and can, therefore, speed up the learning process approximately by A/A' (such that learning is effectively performed on the valid state-action pairs).

Embedding the elimination signal into the MDP is not trivial. One option is to shape the original reward by adding an elimination penalty. That is, decreasing the rewards when selecting the wrong actions. Reward shaping, however, is tricky to tune, may slow the convergence of the function approximation, and is not sample efficient (irrelevant actions are explored). Another option is to design a policy that is optimized by interleaved policy gradient updates on the two signals, maximizing the reward and minimizing the elimination signal error. The main difficulty with this approach is that both models are strongly coupled, and each model affects the observations of the other model, such that the convergence of any of the models is not trivial.

Next, we present a method that decouples the elimination signal from the MDP by using contextual multi-armed bandits. The contextual bandit learns a mapping from states (represented by context vectors $x(s)$) to the elimination signal $e(s,a)$ that estimates which actions should be eliminated. We start by introducing theoretical results on linear contextual bandits, and most importantly, concentration bounds for contextual bandits that require almost no assumptions on the context distribution. We will later show that under this model we can decouple the action elimination from the learning process in the MDP, allowing us to learn using standard Q-learning while eliminating actions correctly.

3.2 Action elimination with contextual bandits

Let $x(s_t) \in \mathbb{R}^d$ be the feature representation of state s_t. We assume (realizability) that under this representation there exists a set of parameters $\theta_a^* \in \mathbb{R}^d$ such that the elimination signal in state s_t is $e_t(s_t,a) = \theta_a^{*T} x(s_t) + \eta_t$, where $\|\theta_a^*\|_2 \leq S$. η_t is an R-subgaussian random variable with zero mean that models additive noise to the elimination signal. When there is no noise in the elimination signal, then $R = 0$. Otherwise, as the elimination signal is bounded in $[0,1]$, it holds that $R \leq 1$. We'll also relax our previous assumptions and allow the elimination signal to have values $0 \leq \mathbb{E}[e_t(s_t,a)] \leq \ell$ for any valid action and $u \leq \mathbb{E}[e_t(s_t,a)] \leq 1$ for any invalid action, with $\ell < u$. Next, we denote by $X_{t,a}$ ($E_{t,a}$) the matrix (vector) whose rows (elements) are the observed state representation vectors (elimination signals) in which action a was chosen, up to time t. For example, the i^{th} row in $X_{t,a}$ is the representation vector of the i^{th} state on which the action a was chosen. Denote the solution to the regularized linear regression $\|X_{t,a}\theta_{t,a} - E_{t,a}\|_2^2 + \lambda\|\theta_{t,a}\|_2^2$ (for some $\lambda > 0$) by $\hat{\theta}_{t,a} = \bar{V}_{t,a}^{-1} X_{t,a}^T E_{t,a}$ where $\bar{V}_{t,a} = \lambda I + X_{t,a}^T X_{t,a}$.

Similarly to Theorem 2 in (Abbasi-Yadkori, Pal, and Szepesvari, 2011)[2], for any state history and with probability of at least $1-\delta$, it holds for all $t > 0$ that $\left|\hat{\theta}_{t,a}^T x(s_t) - \theta_a^{*T} x(s_t)\right| \leq \sqrt{\beta_t(\delta) x(s_t)^T \bar{V}_{t,a}^{-1} x(s_t)}$, where $\sqrt{\beta_t(\delta)} = R\sqrt{2\log(\det(\bar{V}_{t,a})^{1/2}\det(\lambda I)^{-1/2}/\delta)} + \lambda^{1/2}S$. If $\forall s, \|x(s)\|_2 \leq L$, then β_t can be bounded by $\sqrt{\beta_t(\delta)} \leq R\sqrt{d\log(1 + tL^2/\lambda/\delta} + \lambda^{1/2}S$. Next, we

[2] Our theoretical analysis builds on results from (Abbasi-Yadkori, Pal, and Szepesvari, 2011), which can be extended to include General Linear Models (GLMs). We focus on linear contextual bandits as they enjoy easier implementation and tighter confidence intervals in comparison to GLMs. We will later combine the bandit with feature approximation, which will approximately allow the realizability assumption even for linear bandits.

define $\tilde{\delta} = \delta/k$ and bound this probability for all the actions, i.e., $\forall a, t > 0$

$$\Pr\left\{ \left| \hat{\theta}^T_{t-1,a} x(s_t) - {\theta^*_{t-1,a}}^T x(s_t) \right| \le \sqrt{\beta_t(\tilde{\delta}) x(s_t)^T \bar{V}^{-1}_{t-1,a} x(s_t)} \right\} \ge 1 - \delta \tag{1}$$

Recall that any valid action a at state s satisfies $\mathbb{E}[e_t(s,a)] = {\theta^*_a}^T x(s_t) \le \ell$. Thus, we can eliminate action a at state s_t if

$$\hat{\theta}^T_{t-1,a} x(s_t) - \sqrt{\beta_{t-1}(\tilde{\delta}) x(s_t)^T \bar{V}^{-1}_{t-1,a} x(s_t)} > \ell \tag{2}$$

This ensures that with probability $1 - \delta$ we never eliminate any valid action. We emphasize that only the expectation of the elimination signal is linear in the context. The expectation does not have to be binary (while the signal itself is). For example in conversational agents, if a sentence is not understood by 90% of the humans who hear it, it is still desirable to avoid saying it. We also note that we assume ℓ is known, but in most practical cases, choosing $\ell \approx 0.5$ should suffice. In the current formulation, knowing u is not necessary, though its value will affect the overall performance.

3.3 Concurrent Learning

We now show how the Q-learning and contextual bandit algorithms can learn simultaneously, resulting in the convergence of both algorithms, i.e., finding an optimal policy and a minimal valid action space. The challenge here is that each learning process affects the state-action distribution of the other. We first define Action Elimination Q-learning.

Definition 3. *Action Elimination Q-learning is a Q-learning algorithm which updates only admissible state-action pairs and chooses the best action in the next state from its admissible actions. We allow the base Q-learning algorithm to be any algorithm that converges to Q^* with probability 1 after observing each state-action infinitely often.*

If the elimination is done based on the concentration bounds of the linear contextual bandits, we can ensure that Action Elimination Q-learning converges, as can be seen in Proposition 1 (See Appendix A for a full proof).

Proposition 1. *Assume that all state action pairs (s,a) are visited infinitely often, unless eliminated according to $\hat{\theta}^T_{t-1,a} x(s) - \sqrt{\beta_{t-1}(\tilde{\delta}) x(s)^T \bar{V}^{-1}_{t-1,a} x(s)} > \ell$. Then, with a probability of at least $1 - \delta$, action elimination Q-learning converges to the optimal Q-function for any valid state-action pairs. In addition, actions which should be eliminated are visited at most $T_{s,a}(t) \le 4\beta_t/(u-\ell)^2 + 1$ times.*

Notice that when there is no noise in the elimination signal ($R = 0$), we correctly eliminate actions with probability 1. Thus, invalid actions will be sampled a finite (logarithmic under very mild assumptions) number of times.

4 Method

Using raw features like *word2vec*, directly for control, results in exhaustive computations. Moreover, raw features are typically not realizable, i,.e., the assumption that $e_t(s_t, a) = {\theta^*_a}^T x(s_t) + \eta_t$ does not hold. Thus, we propose learning a set of features $\phi(s_t)$ that are realizable, i.e., $e(s_t, a) = {\theta^*_a}^T \phi(s_t)$, using neural networks (using the activations of the last layer as features). A practical challenge here is that the features must be fixed over time when used by the contextual bandit, while the activations change during optimization. We therefore follow a batch-updates framework (Levine et al., 2017; Riquelme, Tucker, and Snoek, 2018), where every few steps we learn a new contextual bandit model that uses the last layer activations of the AEN as features.

Our Algorithm presents a hybrid approach for DRL with Action Elimination (AE), by incorporating AE into the well-known DQN algorithm to yield our AE-DQN (Algorithm 1 and Figure 1(b)). AE-DQN trains two networks: a DQN denoted by Q and an AEN denoted by E. The algorithm uses E, and creates a linear contextual bandit model from it every L iterations with procedure **AENUpdate()**. This procedure uses the activations of the last hidden layer of E as features, $\phi(s) \leftarrow$ LastLayerActivations$(E(s))$, which are then used to create a contextual linear bandit model ($V_a = \sum_{j:a_j=a} \phi(s_j)\phi(s_j)^T, b_a = \sum_{j:a_j=a} \phi(s_j)^T e_j$). AENUpdate() proceeds by solving this model, and plugging the solution into the target AEN (LastLayer(E_a^-) $\leftarrow V_a^{-1} b_a$). The contextual linear bandit model (E^-, V) is then used to eliminate actions (with high probability) via the **ACT()** and **Targets()** functions. ACT() follows an $\epsilon-$greedy mechanism on the admissible actions set $A' = \{a : E(s)_a - \sqrt{\beta \phi(s)^T V_a^{-1} \phi(s)} < \ell\}$. If it decides to exploit, then it selects the action with highest Q-value by taking an $\arg\max$ on Q-values among A', and if it chooses to explore, then, it selects an action uniformly from A'. The **Targets()** procedure is estimating the value function by taking $\max$ over Q-values only among admissible actions, hence, reducing function approximation errors.

Algorithm 1 deep Q-learning with action elimination

```
Input: $\epsilon, \beta, \ell, \lambda, C, L, N$
Initialize AEN and DQN with random weights $\omega, \theta$ respectively, and set target networks $Q^-, E^-$ with a copy of $\theta, \omega$
Define $\phi(s) \leftarrow$ LastLayerActivations($E(s)$)
Initialize Replay Memory D to capacity N
for t = 1,2,..., do
    $a_t$ = ACT($s_t, Q, E^-, V^{-1}, \epsilon, \ell, \beta$)
    Execute action $a_t$ and observe $\{r_t, e_t, s_{t+1}\}$
    Store transition $\{s_t, a_t, r_t, e_t, s_{t+1}\}$ in D
    Sample transitions
        $\{s_j, a_j, r_j, e_j, s_{j+1}\}_{j=1}^m \in D$
    $y_j =$ Targets $\left(s_{j+1}, r_j, \gamma, Q^-, E^-, V^{-1}, \beta, \ell\right)$
    $\theta = \theta - \nabla_\theta \sum_j \left(y_j - Q(s_j, a_j; \theta)\right)^2$
    $\omega = \omega - \nabla_\omega \sum_j \left(e_j - \mathrm{E}(s_j, a_j; \omega)\right)^2$
    If $(t$ mod $C) = 0 : Q^- \leftarrow Q$
    If $(t$ mod $L) = 0 :$
        $E^-, V^{-1} \leftarrow$AENUpdate($E, \lambda, D$)
end for

function ACT($s, Q, E, V^{-1}, \epsilon, \beta, \ell$)
    $A' \leftarrow \{a : E(s)_a - \sqrt{\beta \phi(s)^T V_a^{-1} \phi(s)} < \ell\}$
    With probability $\epsilon$, return Uniform($A'$)
    Otherwise, return $\arg\max_{a \in A'} Q(s, a)$
end function
function TARGETS($s, r, \gamma, Q, E, V^{-1}, \beta, \ell$)
    if $s$ is a terminal state then return $r$ end if
    $A' \leftarrow \{a : E(s)_a - \sqrt{\beta \phi(s)^T V_a^{-1} \phi(s)} < \ell\}$
    return $(r + \gamma \max_{a \in A'} Q(s, a))$
end function
function AENUPDATE($E^-, \lambda, D$)
    for $a \in A$ do
        $V_a^{-1} = \left(\sum_{j:a_j=a} \phi(s_j)\phi(s_j)^T + \lambda I\right)^{-1}$
        $b_a = \sum_{j:a_j=a} \phi(s_j)^T e_j$
        Set LastLayer($E_a^-$) $\leftarrow V_a^{-1} b_a$
    end for
    return $E^-, V^{-1}$
end function
```

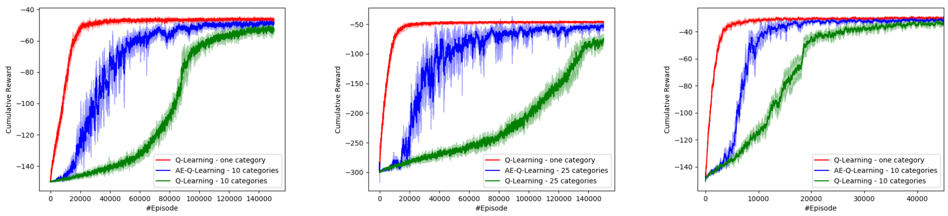
Architectures: The agent uses an Experience Replay (Lin, 1992) to store information about states, transitions, actions, and rewards. In addition, our agent also stores the elimination signal, provided by the environment (Figure 1(b)). The architecture for both the AEN and DQN is an NLP CNN, based on (Kim, 2014). We represent the state as a sequence of words, composed of the game descriptor (Figure 1(a), "Observation") and the player's inventory. These are truncated or zero-padded (for simplicity) to a length of 50 (descriptor) + 15 (inventory) words and each word is embedded into continuous vectors using *word2vec* in $\mathbb{R}^{300}$. The features of the last four states are then concatenated together such that our final state representations s are in $\mathbb{R}^{78,000}$. The AEN is trained to minimize the MSE loss, using the elimination signal as a label. We used 100 (500 for DQN) convolutional filters, with three different 1D kernels of length (1,2,3) such that the last hidden layer size is 300. [3]

5 Experimental Results

Grid World Domain: We started by evaluating our algorithm on a small grid world domain with 9 rooms, where we can carefully analyze the effect of the action elimination (visualization can be found in the appendix). In this domain, the agent starts at the center of the grid and needs to navigate to its upper-left corner. On every step, the agent suffers a penalty of (-1), with a terminal reward of 0. Prior to the game, the states are randomly divided into K categories. The environment has $4K$ navigation actions, 4 for each category, each with a probability to move in a random direction. If the chosen action belongs to the same category as the state, the action is performed correctly in probability $p_c^T = 0.75$. Otherwise, it will be performed correctly in probability $p_c^F = 0.5$. If the action does not fit the state category, the elimination signal equals 1, and if the action and state belong

[3] Our code, the Zork domain, and the implementation of the elimination signal can be found at: `https://github.com/TomZahavy/CB_AE_DQN`

to the same category, then $e = 0$. The optimal policy will only use the navigation actions from the same type as the state, and all of the other actions are strictly suboptimal. A basic comparison between vanilla Q-learning without action elimination (green) and a tabular version of the action elimination Q-learning (blue) can be found on Figure 2. Further details and comparisons can be found in the appendix. In all of the figures, the results are compared to the case with one category (red), i.e., only 4 basic navigation actions, which forms an upper bound on performance with multiple categories. In Figure 2(a),2(c), the episode length is $T = 150$, and in Figure 2(b) it is $T = 300$, to allow sufficient exploration for the vanilla Q-Learning. It is clear from the simulations that the action elimination dramatically improves the results in large action spaces. We also noticed that the gain from action elimination increases with the grid size since the elimination allows the agent to reach the goal earlier.



(a) 30x30,K=1,10

(b) 30x30,K=1,25

(c) 20x20,K=1,10

Figure 2: Performance of agents in grid world.

Zork domain: *"This is an open field west of a white house, with a boarded front door. There is a small mailbox here. A rubber mat saying 'Welcome to Zork!' lies by the door"*. This is an excerpt from the opening scene provided to a player in “Zork I: The Great Underground Empire”; one of the first interactive fiction computer games, created by members of the MIT Dynamic Modeling Group in the late 70s. By exploring the world via interactive text-based dialogue, the players progress in the game. The world of Zork presents a rich environment with a large state and action space (see Figure 3).

Zork players describe their actions using natural language instructions. For example, in the opening excerpt, an action might be ‘open the mailbox’ (Figure 1(a)). Once the player describes his/her action, it is processed by a sophisticated natural language parser. Based on the parser’s results, the game presents the outcome of the action. The ultimate goal of Zork is to collect the Twenty Treasures of Zork and install them in the trophy case. Finding the treasures require solving a variety of puzzles such as the navigation of two complex mazes and intricate action sequences. During the game, the player is awarded points for performing deeds that bring him closer to the game’s goal (e.g., solving

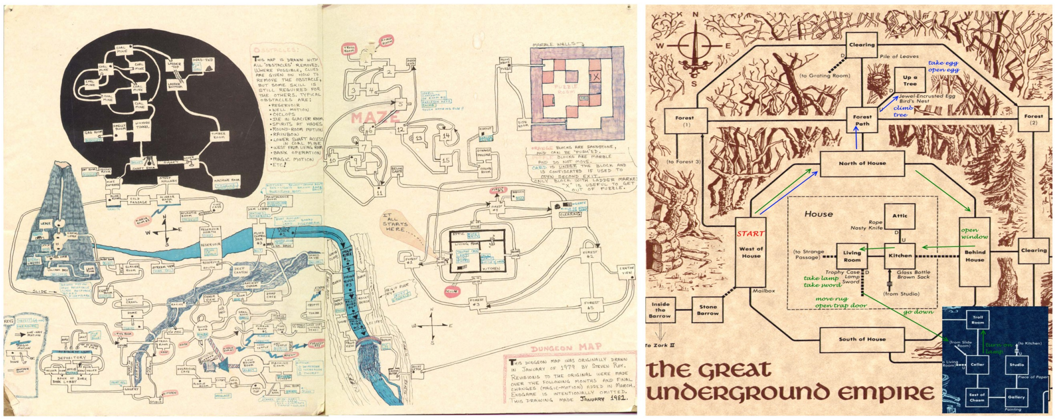
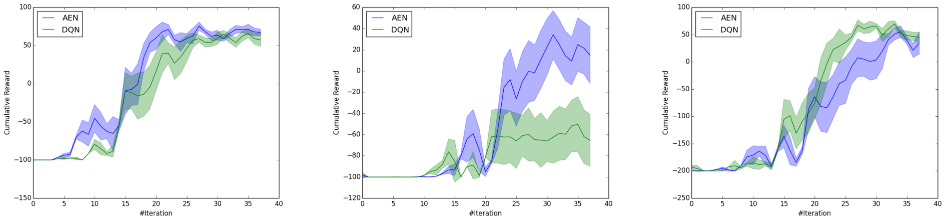


Figure 3: **Left:** the world of Zork. **Right:** subdomains of Zork; the Troll (green) and Egg (blue) Quests. Credit: S. Meretzky, The Strong National Museum of Play. Larger versions in Appendix B.



(a) A_1,T=100

(b) A_2,T=100

(c) A_2,T=200

Figure 4: Performance of agents in the egg quest.

puzzles). Placing all of the treasures into the trophy case generates a total score of 350 points for the player. Points that are generated from the game's scoring system are given to the agent as a reward. Zork presents multiple challenges to the player, like building plans to achieve long-term goals; dealing with random events like troll attacks; remembering implicit clues as well as learning the interactions between objects in the game and specific actions. The **elimination signal** in Zork is given by the Zork environment in two forms, a "wrong parse" flag, and text feedback (e.g. "you cannot take that"). We group these two signals into a single binary signal which we then provide to our learning algorithm.

Before we started experimenting in the "Open Zork" domain, i.e., playing in Zork without any modifications to the domain, we analyzed our algorithm in a small tabular grid world domain. We subsequently evaluated the performance on two subdomains of Zork. These subdomains are inspired by the Zork plot and are referred to as the Egg Quest and the Troll Quest (Figure 3, right, and Appendix B respectively). For these subdomains, we introduced an additional reward signal (in addition to the reward provided by the environment) to guide the agent towards solving specific tasks and make the results more visible (we only use the environment reward when solving "Open Zork"). The optimal time that it takes to solve each quest is 6 in-game timesteps? for the Egg quest, 11 for the Troll quest and 350 for "Open Zork". The agent's goal in each subdomain is to maximize its cumulative reward. A reward of -1 is applied at every time step to encourage the agent to favor short paths. Each trajectory terminates upon completing the quest or after T steps are taken. We set the discounted factor during training to $\gamma = 0.8$ but use $\gamma = 1$ during evaluation [4]. We used $\beta = 0.5, \ell = 0.6$ in all the experiments. The results are averaged over 5 random seeds, shown alongside error bars (std/3) - define more explicitly.

The Egg Quest: In this quest, the agent's goal is to find and open the jewel-encrusted egg, hidden on a tree in the forest. The agent is awarded 100 points upon successful completion of this task. We experimented with the AE-DQN (blue) agent and a vanilla DQN agent (green) in this quest (Figure 4). The action set in this quest is composed of two subsets. A fixed subset of 9 actions that allow it to complete the Egg Quest like *navigate* (south, east etc.) *open* an item and *fight*; And a second subset consists of N_{Take} *"take"* actions for possible objects in the game. The "take" actions correspond to taking a single object and include objects that need to be collected to complete quests, as well as other irrelevant objects from the game dictionary. We used two versions of this action set, A_1 with $N_{\text{Take}} = 200$ and A_2 with $N_{\text{Take}} = 300$.

Robustness to hyperparameter tuning: We can see that for A_1, with T=100, (Figure $4a$), and for A_2, with T=200, (Figure $4c$) *Both* agents solve the task well. However, for A_2, with T=100, (Figure $4b$) the AE-DQN agent learns considerably faster, implying that action elimination is more robust to hyper parameters settings when there are many actions, even in this small quest.

The Troll Quest: In this quest, the agent must find a way to enter the house, grab a lantern and light it, expose the hidden entrance to the underworld and then find the troll, awarding him 100 points. The Troll Quest presents a larger problem than the Egg Quest, but smaller than the full Zork domain; it is large enough to gain a useful understanding of our agents' performance. The AE-DQN (green) and DQN (red) agents use a similar action set to A_1 with 200 take actions and 15 necessary

[4] We adopted a common evaluation scheme that is used in the ALE. During learning and training we use $\gamma < 1$ but evaluation is performed with $\gamma = 1$. Intuitively, during learning, choosing $\gamma < 1$ helps to learn, while during evaluation, the sum of cumulative returns ($\gamma = 1$) is more interpretable (the score in the game).

Table 1: Experimental results in Zork

	#actions	cumulative reward
Kostka et al. 2017	≈ 150	13.5
Ours, A_3	131	**39**
Ours, A_3, 2M steps	131	**44**
Fulda et al. 2017	≈ 500	8.8
Ours, A_4	1227	**16**
Ours, A_4, 2M steps	1227	**16**

actions (215 in total). For comparison, We also included an "optimal elimination" baseline (blue) that consists of only 35 actions (15 essential, and 20 relevant take actions). We can see in Figure 5 that AE-DQN significantly outperforms DQN, achieving compatible performance to the "optimal elimination" baseline. In addition, we can see that the improvement of the AE-DQN over DQN is more significant in the Troll Quest than the Egg quest. This observation is consistent with our tabular experiments.

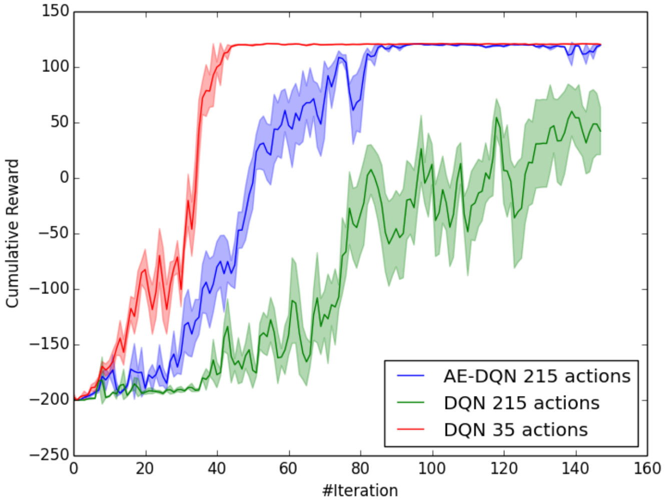


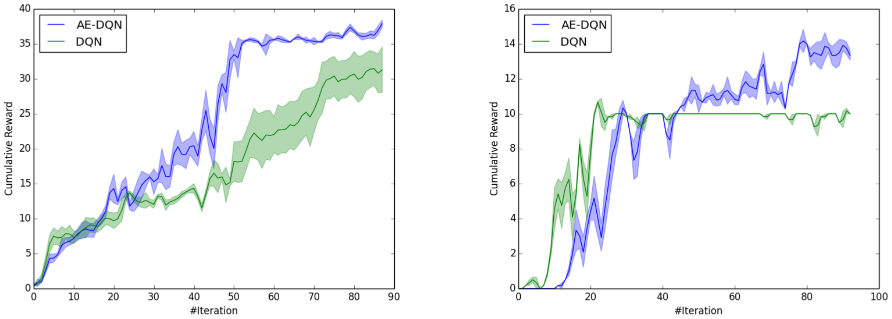
Figure 5: Results in the Troll Quest.

“Open Zork“: Next, we evaluated our agent in the “Open Zork“ domain (without hand-crafting reward and termination signals). To compare our results with previous work, we trained our agent for 1M steps: each trajectory terminates after $T = 200$ steps, and a total of 5000 trajectories were executed [5]. We used two action sets: A_3, the “Minimal Zork“ action set, is the minimal set of actions (131) that is required to solve the game (comparable with the action set used by Kostka et al. (2017)). The actions are taken from a tutorial for solving the game. A_4, the “Open Zork“ action set, includes 1227 actions (comparable with Fulda et al. (2017)). This set is created from action "templates", composed of $\{$Verb, Object$\}$ tuples for all the verbs (19) and objects (62) in the game (e.g, open mailbox). In addition, we include a fixed set of 49 actions of varying length (but not of length 2) that are required to solve the game.

Table 1 presents the maximal reward obtained by our AE-DQN agent in this domain while using action sets A_3 and A_4, showing that our agent achieves state-of-the-art results, outperforming all

[5] The same amount of steps that were used in previous work on Zork (Fulda et al., 2017; Kostka et al., 2017). For completeness, we also report results for AE-DQN with 2M steps, where learning seemed to converge.

previous work. Figure 6 shows the learning curves for both AE-DQN and DQN agents. Again, we can see that AE-DQN outperforms DQN, learning faster and achieving more reward.



(a)

(b)

Figure 6: Results in "Open Zork".

6 Summary

In this work, we proposed the AE-DQN, a DRL approach for eliminating actions while performing Q-learning, for solving MDPs with large state and action spaces. We tested our approach on the text-based game Zork, showing that by eliminating actions, using linear contextual bandits with theoretical guarantees of convergence, the size of the action space is reduced, exploration is more effective, and learning is improved. In future work, we plan to investigate more sophisticated architectures, as well as learning shared representations for elimination and control which may boost performance on both tasks.

In addition, we aim to investigate other mechanisms for action elimination, e.g., eliminating actions that result from low Q-values (Even-Dar, Mannor, and Mansour, 2003). Another direction is to generate elimination signals in real-world domains. This can be done by designing a rule-based system for actions that should be eliminated, and then, training an AEN to generalize these rules for states that were not included in these rules. Finally, elimination signals may be provided implicitly, e.g., by human demonstrations for actions that should not be taken.

References

Abbasi-Yadkori, Y.; Pal, D.; and Szepesvari, C. 2011. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, 2312–2320.

Azizzadenesheli, K.; Brunskill, E.; and Anandkumar, A. 2018. Efficient exploration through bayesian deep q-networks. *arXiv*.

Bartlett, M. S. 1951. An inverse matrix adjustment arising in discriminant analysis. *The Annals of Mathematical Statistics* 22(1).

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Bertsekas, D. P., and Tsitsiklis, J. N. 1995. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, 560–564. IEEE.

Budzianowski, P.; Ultes, S.; Su, P.-H.; Mrksic, N.; Wen, T.-H.; Casanueva, I.; Rojas-Barahona, L.; and Gasic, M. 2017. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint*.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*.

Côté, M.-A.; Kádár, Á.; Yuan, X.; Kybartas, B.; Barnes, T.; Fine, E.; Moore, J.; Hausknecht, M.; Asri, L. E.; Adada, M.; et al. 2018. Textworld: A learning environment for text-based games. *arXiv*.

Dalal, G.; Gilboa, E.; and Mannor, S. 2016. Hierarchical decision making in electricity grid management. In *International Conference on Machine Learning*, 2197–2206.

Dhingra, B.; Li, L.; Li, X.; Gao, J.; Chen, Y.-N.; Ahmed, F.; and Deng, L. 2016. End-to-end reinforcement learning of dialogue agents for information access. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Dulac-Arnold, G.; Denoyer, L.; Preux, P.; and Gallinari, P. 2012. Fast reinforcement learning with large action sets using error-correcting output codes for mdp factorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 180–194. Springer.

Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; and Coppin, B. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv*.

Even-Dar, E., and Mansour, Y. 2003. Learning rates for q-learning. *Journal of Machine Learning Research* 5(Dec):1–25.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2003. Action elimination and stopping conditions for reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning*.

Fulda, N.; Ricks, D.; Murdoch, B.; and Wingate, D. 2017. What can you do with a rock? affordance extraction via word embeddings. *arXiv*.

Glavic, M.; Fonteneau, R.; and Ernst, D. 2017. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine* 6918–6927.

He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2015. Deep reinforcement learning with an unbounded action space. *CoRR* abs/1511.04636.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Sendonaris, A.; Dulac-Arnold, G.; Osband, I.; Agapiou, J.; et al. 2018. Learning from demonstrations for real world reinforcement learning. *AAAI*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kakade, S. M., et al. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation, University of London, England.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint*.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Kostka, B.; Kwiecieli, J.; Kowalski, J.; and Rychlikowski, P. 2017. Text-based adventures of the golovin ai agent. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE.

Lagoudakis, M. G., and Parr, R. 2003. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*.

Lattimore, T., and Hutter, M. 2012. Pac bounds for discounted mdps. In *International Conference on Algorithmic Learning Theory*.

Levine, N.; Zahavy, T.; Mankowitz, D. J.; Tamar, A.; and Mannor, S. 2017. Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 3138–3148.

Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; and Jurafsky, D. 2016. Deep reinforcement learning for dialogue generation. *arXiv*.

Li, X.; Chen, Y.-N.; Li, L.; and Gao, J. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8(3-4).

Lipton, Z. C.; Gao, J.; Li, L.; Chen, J.; and Deng, L. 2016a. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv*.

Lipton, Z. C.; Gao, J.; Li, L.; Li, X.; Ahmed, F.; and Deng, L. 2016b. Efficient exploration for dialogue policy learning with bbq networks and replay buffer spiking. *arXiv*.

Liu, B.; Tur, G.; Hakkani-Tur, D.; Shah, P.; and Heck, L. 2017. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. *arXiv preprint*.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M.; and Bowling, M. 2017. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv*.

Mannion, P.; Duggan, J.; and Howley, E. 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*. Springer. 47–66.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Narasimhan, K.; Kulkarni, T. D.; and Barzilay, R. 2015. Language understanding for text-based games using deep reinforcement learning. *CoRR* abs/1506.08941.

Pazis, J., and Parr, R. 2011. Generalized value functions for large action sets. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1185–1192.

Peng, B.; Li, X.; Li, L.; Gao, J.; Celikyilmaz, A.; Lee, S.; and Wong, K.-F. 2017. Composite task-completion dialogue system via hierarchical deep reinforcement learning. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Riquelme, C.; Tucker, G.; and Snoek, J. 2018. Deep bayesian bandits showdown. *International Conference on Learning Representations*.

Serban, I. V.; Sankar, C.; Germain, M.; Zhang, S.; Lin, Z.; Subramanian, S.; Kim, T.; Pieper, M.; Chandar, S.; Ke, N. R.; et al. 2017. A deep reinforcement learning chatbot. *arXiv preprint*.

Su, P.-H.; Gasic, M.; Mrksic, N.; Rojas-Barahona, L.; Ultes, S.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. Continuously learning neural dialogue management. *arXiv preprint*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM* 58–68.

Thrun, S., and Schwartz, A. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models*.

Van der Pol, E., and Oliehoek, F. A. 2016. Coordinated deep reinforcement learners for traffic light control. In *In proceedings of NIPS*.

Van Hasselt, H., and Wiering, M. A. 2009. Using continuous action spaces to solve discrete problems. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, 1149–1156. IEEE.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Wen, Z.; O'Neill, D.; and Maei, H. 2015. Optimal demand response using device-based reinforcement learning. *IEEE Transactions on Smart Grid* 2312–2324.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint*.

Yuan, X.; Côté, M.-A.; Sordoni, A.; Laroche, R.; Combes, R. T. d.; Hausknecht, M.; and Trischler, A. 2018. Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.

Zahavy, T.; Ben-Zrihem, N.; and Mannor, S. 2016. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, 1899–1908.

Zahavy, T.; Magnani, A.; Krishnan, A.; and Mannor, S. 2018. Is a picture worth a thousand words? a deep multi-modal fusion architecture for product classification in e-commerce. *The Thirtieth Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

Zelinka, M. 2018. Using reinforcement learning to learn how to play text-based games. *arXiv preprint*.

Zhao, T., and Eskenazi, M. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint*.

Zrihem, N. B.; Zahavy, T.; and Mannor, S. 2016. Visualizing dynamics: from t-sne to semi-mdps. *arXiv preprint arXiv:1606.07112*.

Appendix A Proof of Proposition 1

Proposition 1. *Assume that all state action pairs (s, a) are visited infinitely often, unless eliminated according to $\hat{\theta}_{t-1,a}^T x(s) - \sqrt{\beta_{t-1}(\tilde{\delta}) x(s)^T \bar{V}_{t-1,a}^{-1} x(s)} > \ell$. Then, with a probability of at least $1-\delta$, action elimination Q-learning converges to the optimal Q-function for any valid state-action pairs. In addition, actions which should be eliminated are visited at most $T_{s,a}(t) \leq 4\beta_t/(u-\ell)^2 + 1$ times.*

Proof. We start by proving the convergence of the algorithm and then prove the bound on the number of visits of invalid actions.

Denote the MDP as M. According to Equation 1, with probability of at least $1-\delta$, elimination by Equation 2 never eliminates a valid action, and thus all of these actions are visited infinitely often. If all of the state-action pairs are visited infinitely often even after the elimination, the Q-learning will converge at all state-action pairs. Otherwise, there are some invalid actions, which are strictly suboptimal, and are visited a finite number of times. In this case, there exists some time $T < \infty$ such that all of these actions are never played for any $t > T$. Define a new MDP $\tilde{M}$, as M without any of the eliminated actions. As these actions are strictly suboptimal, the value of $\tilde{M}$ will be identical to the value of M in all states, and so are the Q-values for any action that survived the elimination. Furthermore, $\tilde{M}$ contains all of the valid states, and their Q-values will be identical those of M, as they only depend on the reward in the valid state-action pairs and the value in the next state, both which exist in $\tilde{M}$. For any $t > T$, M is equivalent to $\tilde{M}$, and all of its state-actions are visited infinitely often. Therefore, the Q-function will converge to the optimal Q-function with probability 1 in all of $\tilde{M}$'s state-action pairs. Specifically, it will converge in all of valid state-action pairs (s, a), which concludes the first part of the proof.

We'll now prove the sample complexity of any invalid actions. First, note that the confidence bound is strongly related to the number of visits in a state-action pair:

$$\begin{aligned}
x(s_t)^T \bar{V}_{t-1,a}^{-1} x(s_t) &= x(s_t)^T \left\{ \lambda I + T_{s,a}(t-1) x(s_t) x(s_t)^T + \sum_{s' \neq s_t} T_{s',a}(t-1) x(s') x(s')^T \right\}^{-1} x(s_t) \\
&\overset{(1)}{\leq} x(s_t)^T \left\{ \lambda I + T_{s,a}(t-1) x(s_t) x(s_t)^T \right\}^{-1} x(s_t) \\
&\overset{(2)}{=} \frac{\|x(s_t)\|^2}{\lambda} - \frac{T_{s,a}(t-1) \frac{\|x(s_t)\|^4}{\lambda^2}}{1 + T_{s,a}(t-1) \frac{\|x(s_t)\|^2}{\lambda}} \\
&= \frac{\|x(s_t)\|^2}{\lambda + T_{s,a}(t-1) \|x(s_t)\|^2} \leq \frac{1}{T_{s,a}(t-1)}
\end{aligned} \tag{3}$$

(1) is correct due to the fact that for any positive definite A and positive semidefinite B, the difference $A^{-1} - (A+B)^{-1}$ is positive semidefinite. (2) is correct due to the Sherman–Morrison formula (Bartlett, 1951). We note that this bound is not tight because it does not use the correlations between different contexts. In fact, the same bound can be probably achieved by placing a multi-armed bandit algorithm in each state. Deriving a tighter bound that utilizes the correlation between contexts is hard, as it is possible to observe a state that its context is not correlated with other states' contexts. Nevertheless, the confidence bounds for contextual bandits can be used in the non-tabular case, in contrast to a MAB formulation.

This implies that a satisfactory condition for correct elimination is

$$\begin{aligned}
&x(s_t)^T \hat{\theta}_{t-1,a} - \sqrt{\beta_{t-1}(\tilde{\delta}) x(s_t)^T \bar{V}_{t-1,a}^{-1} x(s_t)} \\
&\qquad\qquad \overset{(1)}{\geq} u - 2\sqrt{\beta_{t-1}(\tilde{\delta}) x(s_t)^T \bar{V}_{t-1,a}^{-1} x(s_t)} \overset{(2)}{\geq} u - 2\sqrt{\frac{\beta_{t-1}(\tilde{\delta})}{T_{s,a}(t-1)}} > \ell
\end{aligned}$$

where (1) is correct due to Equation 2 with $\mathbb{E}[e(s_t, a)] = \theta_a^{*T} x(s_t) \geq u$, with probability $1 - \delta$, and (2) is correct due to Equation 3. Therefore, if $T_{s,a}(t) \geq 4\frac{\beta_t}{(u-\ell)^2}$ then action a in state s is correctly eliminated. We emphasize that the bound does not depend on the algorithm that chooses state-actions, except for the dependency of β_t, through $\bar{V}_{t,a}$, in the history. Using the fact that β_t is monotonically increasing with t, with probability $1 - \delta$, all of the invalid actions are sampled no more than

$$\begin{aligned} T_{s,a}(t) &\leq \sum_{\tau=1}^{t} \mathbb{1}\left\{T_{s,a}(\tau) \leq 4\frac{\beta_\tau}{(u-\ell)^2}\right\} \\ &\leq \sum_{\tau=1}^{t} \mathbb{1}\left\{T_{s,a}(\tau) \leq 4\frac{\beta_t}{(u-\ell)^2}\right\} \leq 4\frac{\beta_t}{(u-\ell)^2} + 1 \end{aligned}$$

If the sub-gaussianity parameter is $R = 0$, we have $\beta_t = \lambda S^2 < \infty$, and therefore an arm will be sampled at most a finite number of times $T_0 = 4\frac{\lambda S^2}{(u-\ell)^2} + 1 < \infty$. Otherwise, if the state representations are bounded, i.e. $\forall s, \|x(s)\|_2 \leq L$, then, using the simpler form of β_t, the bound can be written as $\lim_{t\to\infty} \frac{T_{s,a}(t)}{\log\left(\frac{t}{\delta}\right)} \leq \frac{4R^2 d}{(u-\ell)^2}$, which means an invalid action is sampled a logarithmic number of times.

□

Appendix B Additional details for grid world simulation

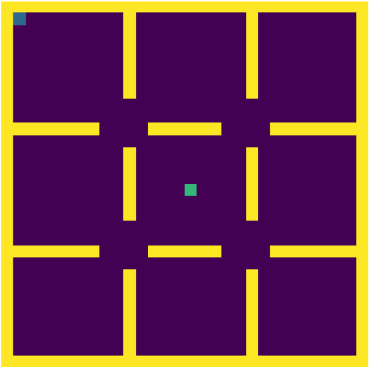


Figure 7: 30x30 Grid World - the agent starts at the center and needs to navigate to the upper-left corner.

An example of a 30x30 grid world map can be found in Figure 7.

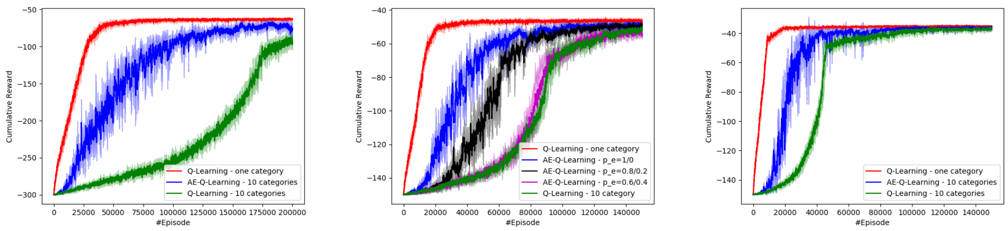
Throughout this secion, we will analyze the effect of the domain's parameters on the performance of the action elimination by changing one parameter at a time, from the following default configuration:

- The default grid size is 30x30
- The number of state categories is $K = 10$
- If the chosen action is from the same category as the current state, it is performed correctly in probability $p_c^T = 0.75$, and if the state and action types are different, the probability is $p_c^F = 0.25$
- We will slightly expend the domain description in the main paper and allow the elimination signal to be random - if the action does not fit the state category, an elimination signal will be equal 1 in probability p_e^F. If the action and state belong to the same category, the probability is p_{e_T}.

- Unless mentioned otherwise, the maximal episode length is $T = 150$.

On each of the simulations, the results were filtered by a moving average filter of length 200, which is needed due to the stochastic nature of the domain. The results are averaged over 5 random seeds, shown alongside error bars (std/3).

Since the problem is tabular, we use confidence intervals in the spirit of UCT (Kocsis and Szepesvári, 2006) - denote the empirical mean of the elimination signal by $\bar{e}(s,a)$ and the number of visits in a state-action pair by $N(s,a)$. An action will be eliminated if $\bar{e}(s,a) - \sqrt{\frac{2\sum_a N(s,a)}{N(s,a)}} > \ell \triangleq 0.5$. The Q-function was initially set to 0, and the learning rates were chosen according to (Even-Dar and Mansour, 2003). The results can be found in Figure 8.



(a) 40x40,K=1,10, T=300

(b) 30x30,different elimination probability

(c) 30x30, different random step probability

Figure 8: Performance of agents in grid world.

Figure 8(a) present the simulation results with a 40x40 grid world. When compared to Figures 2(a),2(c), with grid of 30x30 and 20x20, respectively, we can conclude that action elimination becomes more effective as the grid size increases. Intuitively, it is relatively easy to reach the goal state on small grids, even if the action space is large, and therefore even random exploration will bring the agent to the goal quickly. From the moment the agent reaches the goal, its policy will be biased towards the goal's direction, and it becomes easier to distinguish between valid and invalid actions.

Figure 8(b) shows the performance of the action elimination in cases where eliminating actions is harder, due to randomness in the elimination signal. Specifically, we let p_e^F change between 1 to 0.6 in invalid actions, and p_e^T between 0 to 0.4 in valid actions. We observe that only when the elimination signal is almost completely random, the action elimination algorithm does not present superior performance. Similarly, Figure 8(c) present a scenario where valid actions has almost no randomness ($p_c^T = 0.9$), while invalid actions are almost completely random ($p_c^F = 0.1$). Thus, it is easier to identify the invalid actions, and specifically, understand that these actions are suboptimal. The results indeed show that while action elimination Q-learning converges faster than the vanilla Q-learning, the difference between the convergence rates is smaller.

In summary, the tabular simulations showed a significant improvement due to action elimination, especially when the action space is large, the optimal and suboptimal actions are hard to distinct and the horizon required to reach the goal is large.

Appendix C Maps of Zork

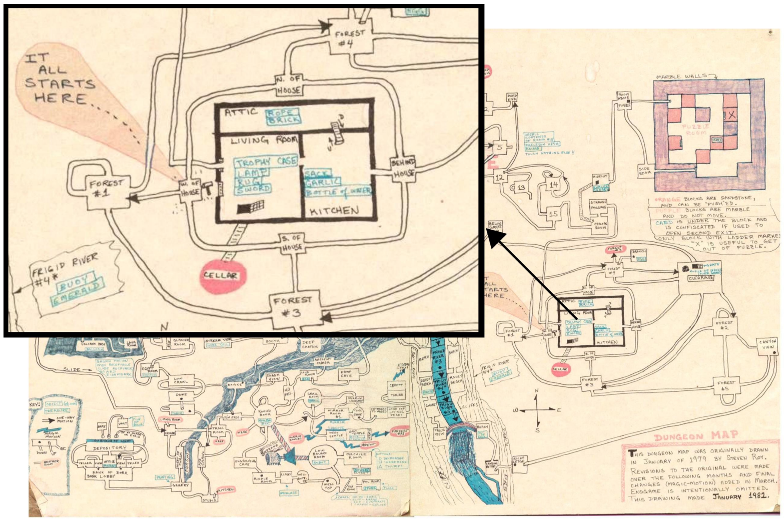

FOREST #4
IT ALL STARTS HERE...
N. OF HOUSE
ATTIC
ROPE
BRICK
LIVING ROOM
TROPHY CASE
LAMP
RUG
SWORD
SACK
GARLIC
BOTTLE of WATER
KITCHEN
BEHIND HOUSE
FOREST #1
W. OF HOUSE
S. OF HOUSE
CELLAR
FRIGID RIVER #4*
RUBY
EMERALD
FOREST #3
N
DUNGEON MAP
THIS DUNGEON MAP WAS ORIGINALLY DRAWN IN JANUARY OF 1979 BY STEVEN ROY. REVISIONS TO THE ORIGINAL WERE MADE OVER THE FOLLOWING MONTHS AND FINAL CHANGES (MAGIC-MOTION) ADDED IN MARCH. ENDGAME IS INTENTIONALLY OMITTED. THIS DRAWING MADE JANUARY 1982.

Figure 9: The world of Zork

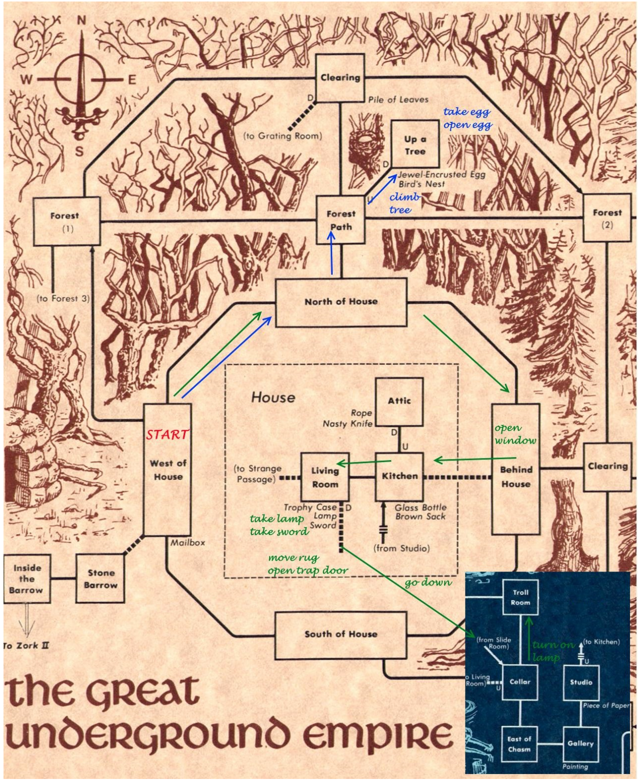


Figure 10: Subdomains of Zork; the Troll (green) and Egg (blue) Quests. Credit: S. Meretzky, The Strong National Museum of Play.