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

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 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 very large** 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 & 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 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 obviously inferior, thus wasting computation time. Control systems for large industrial processes like power grids (Wen et al., 2015; Glavic et al., 2017; Dalal et al., 2016) and traffic control (Mannion et al., 2016; Van der Pol & 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 & 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 et al., 2015; He et al., 2015; Zelinka, 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. While most of the research in the RL community has been focused on dealing with large state spaces, there has been less attention in the literature with regards to large discrete action spaces. Most of the prior work concentrated on factorizing the action space into binary subspaces (Pazis & Parr, 2011; Dulac-Arnold et al., 2012; Lagoudakis & Parr, 2003). Other works proposed to embed the discrete actions into a continuous space. Then, they use a continuous-action policy gradient to find optimal actions in the continuous space and choose the nearest discrete action (Dulac-Arnold et al., 2015; Van Hasselt & Wiering, 2009). He et al. (2015) extended Deep Q-Networks (DQNs, Mnih et al. (2015)) to unbounded action spaces by learning action representations and then choosing the action that provides the highest Q value. However, they only considered large action spaces where a small number of actions (4) are present in each state.

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. 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, and then, designing a machine learning algorithm that will generalize among these rules. 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 should be easier to predict which actions are invalid or obviously inferior in each state and leverage that information for control, rather than learning the actual Q function for all possible state-action pairs. We provide an argument to support this assumption in Section 3.

More specifically, we propose a system that learns an ap-

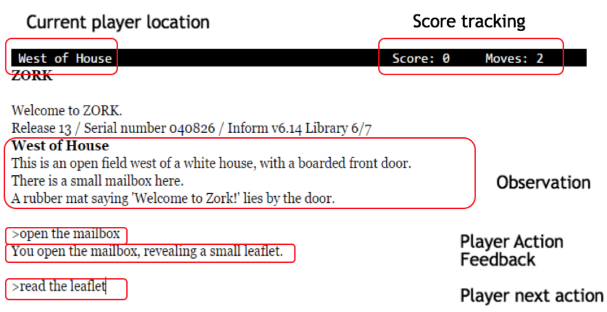


Figure 1. Zork interface

proximation of the Q-function, and concurrently learns to eliminate actions. We focus on tasks where natural language characterizes both the states and the actions, which increases the complexity of the problem since, in addition to eliminating irrelevant actions, good representations of the states and actions are necessary. We introduce a novel DRL approach with two networks, a DQN and an Action Elimination Network (AEN), both designed using a Convolutional Neural Network (CNN) that is suited to NLP tasks (Kim, 2014). The AEN learns to eliminate irrelevant actions, and the DQN learns Q-values for the remaining 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). The player can type in any command, corresponding to an in-game action. Since the input is text-based, this yields more than a thousand of possible actions in each state (e.g., "open door", "open mailbox", "close door" 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): Before the ubiquitousness of graphical displays, text-based games like Zork were popular in the adventure gaming and role-playing communities. Such games propose many challenges for AI research [1]. These include *stochastic dynamics*, *delayed consequences* - actions that have long term consequences; *memory* - the agent may have to remember which actions it took in the past; *dealing with inventory* - items can be stored in an inventory to be used at a later stage; *action selection* - there are many actions to choose from in each state as the agent interacts with the environment using natural language (See Figure 1). In addition, some TBGs introduce stochastic dynamics. For example, in Zork, with random probability a troll can kill you, a thief can appear in each room, and the inventory may get full. Stochasticity is challenging for DRL agents and is currently missing in standard benchmarks (Machado et al., 2017) like the Arcade Learning Environment.

[1]See The CIG Competition for General Text-Based Adventure Game Playing Agents, `http://atkrye.github.io/IEEE-CIG-Text-Adventurer-Competition/`

Representations for text: 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 large variety of text 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) is the default choice (Kim, 2014).

Representations for TBG: 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 (LSTM, Hochreiter & Schmidhuber (1997)) to learn a representation end-to-end, and Zelinka (2018), combined these two approaches.

Action Elimination: Learning to eliminate actions was first mentioned by Even-Dar et al. (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 dangerous 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 & 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 & 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 use the following definitions throughout the paper:

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 et al., 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 & 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 (for valid actions) 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 et al., 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 & 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, speedup 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 bad 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 in this approach is that both models are strongly coupled, and each model affects the observations of the other model, such that 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-sub-Gaussian 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 $\bar{V}_{t,a} = \lambda I + X_{t,a}^T X_{t,a}$.

Similar to Theorem 2 in (Abbasi-Yadkori et al., 2011), 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 $\sqrt{\beta_t(\delta)} = R\sqrt{2\log(\frac{\det(\bar{V}_{t,a})^{1/2}\det(\lambda I)^{-1/2}}{\delta})} + \lambda^{1/2}S$. If $\forall s, \|x(s)\|_2 \leq L$, then β_t can be bounded by $\sqrt{\beta_t(\delta)} \leq R\sqrt{d\log(\frac{1+tL^2/\lambda}{\delta})}+\lambda^{1/2}S$. Next, we define $\tilde{\delta}=\frac{\delta}{k}$ and bound this probability for all the actions, i.e., $\forall a, t>0$ we get that

$$\Pr\left\{\left|\hat{\theta}_{t-1,a}^{T}x(s_t)-{\theta_{t-1,a}^{*}}^{T}x(s_t)\right| \leq \sqrt{\beta_t(\tilde{\delta})x(s_t)^T\bar{V}_{t-1,a}^{-1}x(s_t)}\right\} \geq 1-\delta \quad (1)$$

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

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

This ensures that with probability $1-\delta$ we never eliminate any valid action. Notice that when there is no noise in the elimination signal ($R=0$), we correctly eliminate actions with probability 1.

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, which we address below, 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.*

Given the contextual bandit action elimination result, we can ensure that Action Elimination Q-learning converges by 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 $\hat{\theta}_{t-1,a}^{T}x(s)-\sqrt{\beta_{t-1}(\tilde{\delta})x(s)^T\bar{V}_{t-1,a}^{-1}x(s)} > \ell$. Then, with 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 $T_{s,a}(t) \leq 4\frac{\beta_t}{(u-\ell)^2}+1$ times.*

Note that in the noiseless case ($R=0$), invalid actions will be sampled a finite number of times, and otherwise, under very mild assumptions, a logarithmic number of times.

In practice, the assumption that $e_t(s_t,a)=\theta_a^{*T}x(s_t)+\eta_t$ does not hold for raw features like word2vec. In addition, the elimination signal is usually deterministic, which results in $R=0$ and a constant β which makes the solution less robust to noise in the features. We believe this issue can be solved by learning features $\phi(s_t)$ that are realizable, i.e., $e(s_t,a)=\theta_a^{*T}\phi(s_t)$, for example using neural networks. Nevertheless, doing so in practice is not trivial, as the features must be fixed when used by the contextual bandit.

Algorithm 1 deep Q-learning with action elimination

```
Input: ε, τ, C, N, p, n_sample, n_max
Initialize Elimination and Q Networks with random weights ω, θ respectively
Initialize Target Q Network Q⁻ with a copy of θ
Initialize an empty Replay Memory D to capacity N
for t = 1,2,..., do
  a_t = ACT(s_t, Q, E, ε, n_sample, n_max)
  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 minibatch of transitions {s_j, a_j, r_j, e_j, s_{j+1}} from D
  y_j = Targets(s_{j+1}, r_j, γ, Q⁻, E, τ)
  Perform a gradient descent step on (y_j − Q(s_j, a_j; θ))^2
  Perform a gradient descent step on BCE(e_j, E(s_j, a_j; ω))
  if (t mod C) = 0 then
    Q⁻ ← Q
  end if
end for
```

```
Function Act (s, Q, E, p, ε, n_sample, n_max, τ):
  E_prediction ← E(s, a)
  With probability ε, return Explore(A, E_prediction, p, τ)
  Otherwise,
  A' ← top_{n_max} {E_prediction} ∪ {Mult(E_perdiction)}_{i=1}^{n_sample}
  return arg max_{a'∈A'} Q(s, a)
Function Explore (A, E_perdiction, p, τ):
  WHILE(True) do:
    a ← Uniform(|A|)
    If E_prediction[a] < τ then return a
    Otherwise, with probability p return a
  EndWhile
Function Targets (s, r, γ, Q, E, τ):
  If s is terminal then return r
  Otherwise,
    A' ← {a : E(s, a) ≤ τ}
    return (r + γ max_{a'∈A'} Q(s, a))
```

4. Method

While the previous section provided theoretical guarantees for action elimination using contextual bandits, in practice, it is not clear which features to use. Raw features like word2vec are too high dimensional, resulting in exhaustive computations. Features that are being learned by a DNN (e.g., the activation of the last layer of the AEN or DQN) are not fixed over time. Nevertheless, we now present an approximate solution that is motivated by theory, i.e., eliminating actions with high probability. We leave the empirical integration of the contextual bandits using neural networks to future work (more on that in Section 6).

Algorithm: We now present 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 2). AE-DQN trains two networks: a DQN denoted by Q and an AEN denoted by E. Action elimination is used by the AE-DQN using the following three procedures: **(1) ACT() -** selecting the action with highest Q-value by taking an $\arg\max$ on Q-values among admissible actions A'. The subset A' is generated at each time step and consists of the $n_{\textbf{max}}$ most likely valid actions (sorted according to the AEN probabilities) and an additional $n_{\textbf{sample}}$ actions that are drawn at random from a multinomial distribution w.p. softmax$(1 - \text{prediction})$ (similar to Boltzmann exploration, but on the AEN predictions). This sampling procedure implicitly prioritizes actions by the confidence of the AEN in eliminating them. We assume that there are at most $n_{\textbf{valid}}$ valid actions at each state, and that $n_{\textbf{max}} \geq n_{\textbf{valid}}$. **(2) Explore() -** giving a higher probability to admissible actions, i.e., by adjusting an ϵ-greedy algorithm to give a higher probability to these actions. **(3) Targets() -** estimating the value function by taking $\max$ over Q-values only among admissible actions, hence, reducing function approximation errors. The *Targets()* procedure defines actions as admissible if their predictions are smaller than some threshold τ; this reduces the effect of using invalid actions during bootstrapping. **Architectures:** The agent uses an Experience Replay (Lin, 1992) to store information about states, transitions, actions and rewards. In addition, our agent also stores feedback from the emulator regarding the validity of its actions. Based on this information, we designed an NLP CNN classification architecture, based on (Kim, 2014), to predict actions' relevance in each state. We represent the state as a sequence of words, composed of the game descriptor (Figure 1, "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 (Mikolov et al., 2013) 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 BCE loss (binary cross-entropy) over all possible game actions and estimates the probabilities for actions to fail in a given state. We used 20 convolutional filters, with three different 1D kernels of length (1,2,3) such that the last hidden layer size is 60. Our DQN uses the same network architecture but with 500 filters (last layer size 1500). [2]

[2]Our code, the Zork domain, and the implementation of the elimination signal can be found at: http://anonymous.4open.science/repository/5cc8d217-347e-4307-a895-a13dc972df75/

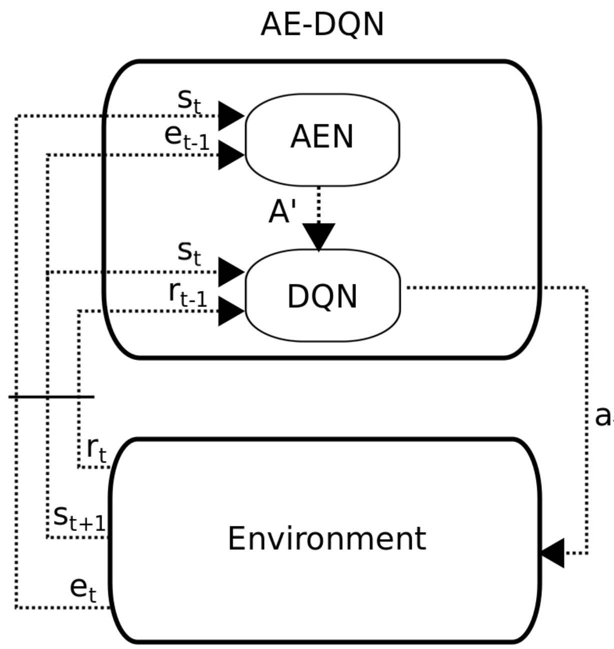


Figure 2. Diagram for AE-DQN.

5. Experimental Results

"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 of the opening 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). 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 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

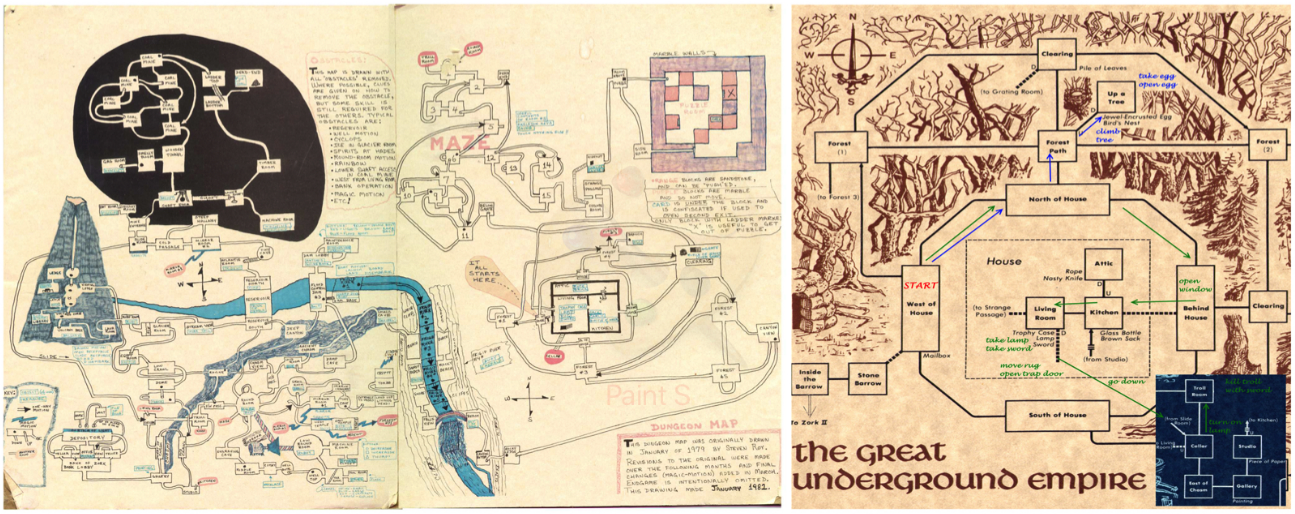


Figure 3. **Left:** the world of Zork. **Right:** subdomains of Zork; the Troll (green) and Egg (blue) Quests. Credit: S. Meretzky, The Strong National Museum of Play. Larger versions in Appendix B.

objects in the game and specific actions.

Before we started experimenting in the "Open Zork" domain, i.e., playing in Zork without any manipulations on the domain, we evaluated our algorithm in two subdomains of Zork. These subdomains are inspired by the Zork plot and referred to as the Egg Quest and the Troll Quest (Figure 3, right, and Appendix B). 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 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 $T = 100$ steps are taken. We set the discounted factor γ during training to $\gamma = 0.8$ but use $\gamma = 1$ during evaluation (as in the DQN paper).

We considered three different action spaces. (1) The "take" action set, 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*. A similar set is used in the Troll Quest, but with 15 actions. The second subset consists of 200 *"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 in order to complete quests, as well as other irrelevant objects from the game dictionary. (2) The "Minimal Zork" action set, is the minimal set of actions (131) that is required to solve the game. The actions are taken from a tutorial for solving the game. (3) The "Open Zork" action set, includes 1227 actions. 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.

5.1. The Egg Quest: Action Set Size

In this quest, the agent's goal is to find and open the jewel-encrusted egg, hidden up 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(a)). The goal of this experiment is to test the effect that the size of the action set has on learning. For that goal, we experimented with two action sets: action set 1 [3] and action set 3 [4]. We can see that for action set 1 (Figure 4(a), top), *Both* agents can solve the task; however, the AE-DQN agent learns considerably faster. Increasing the number of actions to action set 3 (Figure 4(a), bottom) makes it impossible for the vanilla agent to solve the task. On the other hand, the AE-DQN was able to solve it, implying that action elimination is crucial for large action spaces.

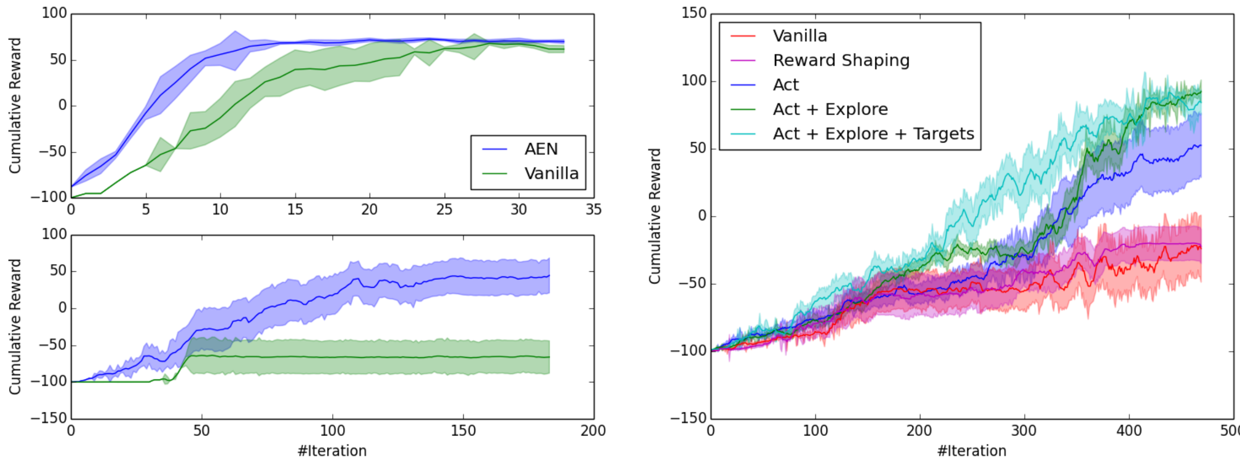
5.2. The Troll Quest: Ablative Analysis

In this quest, the agent must find a way to enter the house, grab a sword and a lantern, expose the hidden entrance to the underworld and then defeat the troll guarding it, 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 agent uses action set (1) consisting of 200 take actions and 15 essential actions (215 in total).

Figure 4(b) presents an ablative analysis of our method in the Troll quest, where we chose $n_{\text{max}} = 10, n_{\text{sample}} = 5$ for the AE mechanism. We can see that the vanilla agent struggles to solve this quest. A second baseline, termed reward shaping, represents a vanilla agent trained with an additional reward of -1 that is given after an invalid action was taken (similar to (Lipton et al., 2016a)). We can see that although

[3] contains 209 actions, where $n_{\text{max}} = 5, n_{\text{sample}} = 2$ were chosen for the AE mechanism

[4] contains 1227 actions, where $n_{\text{max}} = 100, n_{\text{sample}} = 20$ were chosen for the AE mechanism



(a) Egg quest, with action sets (1) (top) and (3) (bottom)

(b) Troll quest, 209 actions, ablative analysis

Figure 4. A comparison of different agents performance in sub-domains of Zork. Results are averaged over 5 random seeds and are shown alongside error bars (std/3).

the reward shaping improves the agent's performance, it is still low. Next, we present three variants of our approach. The first, *act*, uses elimination only for the $ACT()$ procedure and performs standard epsilon-greedy exploration (Act procedure, Algorithm 1). The second, *act+explore*, uses our exploration mechanism in addition to *act* (Explore procedure, Algorithm 1), and the third *act+explore+targets* also uses elimination for training (Targets procedure, Algorithm 1). We can see that AE allows the agent to solve the quest and that adding each of the components provides additional improvement.

5.3. "Open Zork"

Next, we evaluated our agent in the "Open Zork" domain. To compare our results with previous work, we trained our agent for 1M steps: each trajectory terminates upon completing $T = 500$ steps, and a total of 2000 trajectories were executed [5]. We used two action sets: action set (2), which was created from the game tutorial, is comparable with the one used by Kostka et al. (2017); Action set (3), which contains all verb-noun tuples, is comparable with (Fulda et al., 2017) (it is larger but uses more prior knowledge on the domain). Table 1 presents the maximal reward obtained by our AE-DQN agent in this domain while using action sets 2&3, showing that our agent achieves state-of-the-art results, outperforming all previous work.

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, 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 action elimination and control which may boost performance on both tasks (Jaderberg et al., 2016).

Our theoretical analysis in Section 3 suggests that using linear contextual bandits for action elimination guarantees convergence in high probability. In practice, the features that are learned by the AEN can be used for learning a linear contextual bandit on top of the representation of the last layer of the AEN. Since these features must be fixed to be used for learning the bandit, in future work we plan on pursuing a shallow update approach; i.e., continuously training an AEN with new data in order to learn good representation, but every few steps, learning a bandit model on top of it to gain accurate uncertainty estimates and better exploration (Levine et al., 2017; Azizzadenesheli et al., 2018; Riquelme et al., 2018).

In addition, we aim to investigate other mechanisms for action elimination, e.g., eliminating actions that result from low Q-values (Even-Dar et al., 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.

Table 1. Experimental results in Zork

	#actions	cumulative reward
Kostka et al. (2017)	≈ 150	13.5
Ours, action set 2	131	**39**
Fulda et al. (2017)	≈ 500	8.8
Ours, action set 3	1227	**16**

[5] The same amount of steps that was used in previous work on Zork (Fulda et al., 2017; Kostka et al., 2017).

References

Abbasi-Yadkori, Yasin, Pal, David, and Szepesvari, Csaba. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

Azizzadenesheli, Kamyar, Brunskill, Emma, and Anandkumar, Animashree. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

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

Budzianowski, Pawel, Ultes, Stefan, Su, Pei-Hao, Mrksic, Nikola, Wen, Tsung-Hsien, Casanueva, Inigo, Rojas-Barahona, Lina, and Gasic, Milica. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210*, 2017.

Chu, Wei, Li, Lihong, Reyzin, Lev, and Schapire, Robert. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dalal, Gal, Gilboa, Elad, and Mannor, Shie. Hierarchical decision making in electricity grid management. In *International Conference on Machine Learning*, pp. 2197–2206, 2016.

Dhingra, Bhuwan, Li, Lihong, Li, Xiujun, Gao, Jianfeng, Chen, Yun-Nung, Ahmed, Faisal, and Deng, Li. End-to-end reinforcement learning of dialogue agents for information access. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2016.

Dulac-Arnold, Gabriel, Denoyer, Ludovic, Preux, Philippe, and Gallinari, Patrick. 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*, pp. 180–194. Springer, 2012.

Dulac-Arnold, Gabriel, Evans, Richard, van Hasselt, Hado, Sunehag, Peter, Lillicrap, Timothy, Hunt, Jonathan, Mann, Timothy, Weber, Theophane, Degris, Thomas, and Coppin, Ben. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Even-Dar, Eyal, Mannor, Shie, and Mansour, Yishay. Action elimination and stopping conditions for reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 162–169, 2003.

Fulda, Nancy, Ricks, Daniel, Murdoch, Ben, and Wingate, David. What can you do with a rock? affordance extraction via word embeddings. *arXiv preprint arXiv:1703.03429*, 2017.

Glavic, Mevludin, Fonteneau, Raphael, and Ernst, Damien. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, pp. 6918–6927, 2017.

He, Ji, Chen, Jianshu, He, Xiaodong, Gao, Jianfeng, Li, Lihong, Deng, Li, and Ostendorf, Mari. Deep reinforcement learning with an unbounded action space. *CoRR*, abs/1511.04636, 2015. URL `http://arxiv.org/abs/1511.04636`.

Hester, Todd, Vecerik, Matej, Pietquin, Olivier, Lanctot, Marc, Schaul, Tom, Piot, Bilal, Sendonaris, Andrew, Dulac-Arnold, Gabriel, Osband, Ian, Agapiou, John, et al. Learning from demonstrations for real world reinforcement learning. *AAAI*, 2018.

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

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z, Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kakade, Sham Machandranath et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

Kim, Yoon. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Kostka, Bartosz, Kwiecieli, Jaroslaw, Kowalski, Jakub, and Rychlikowski, Pawel. Text-based adventures of the golovin ai agent. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, pp. 181–188. IEEE, 2017.

Lagoudakis, Michail G and Parr, Ronald. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 424–431, 2003.

Lattimore, Tor and Hutter, Marcus. Pac bounds for discounted mdps. In *International Conference on Algorithmic Learning Theory*, pp. 320–334. Springer, 2012.

Levine, Nir, Zahavy, Tom, Mankowitz, Daniel J, Tamar, Aviv, and Mannor, Shie. Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3138–3148, 2017.

Li, Jiwei, Monroe, Will, Ritter, Alan, Galley, Michel, Gao, Jianfeng, and Jurafsky, Dan. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Li, Xuijun, Chen, Yun-Nung, Li, Lihong, and Gao, Jianfeng. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

Lin, Long-Ji. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Lipton, Zachary C, Gao, Jianfeng, Li, Lihong, Chen, Jianshu, and Deng, Li. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016a.

Lipton, Zachary C, Gao, Jianfeng, Li, Lihong, Li, Xiujun, Ahmed, Faisal, and Deng, Li. Efficient exploration for dialogue policy learning with bbq networks and replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 2016b.

Liu, Bing, Tur, Gokhan, Hakkani-Tur, Dilek, Shah, Pararth, and Heck, Larry. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. *arXiv preprint arXiv:1711.10712*, 2017.

Machado, Marlos C, Bellemare, Marc G, Talvitie, Erik, Veness, Joel, Hausknecht, Matthew, and Bowling, Michael. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv preprint arXiv:1709.06009*, 2017.

Mannion, Patrick, Duggan, Jim, and Howley, Enda. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pp. 47–66. Springer, 2016.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Narasimhan, Karthik, Kulkarni, Tejas D., and Barzilay, Regina. Language understanding for text-based games using deep reinforcement learning. *CoRR*, abs/1506.08941, 2015. URL `http://arxiv.org/abs/1506.08941`.

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

Peng, Baolin, Li, Xiujun, Li, Lihong, Gao, Jianfeng, Celikyilmaz, Asli, Lee, Sungjin, and Wong, Kam-Fai. Composite task-completion dialogue system via hierarchical deep reinforcement learning. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Riquelme, Carlos, Tucker, George, and Snoek, Jasper. Deep bayesian bandits showdown. *International Conference on Learning Representations (ICLR)*, 2018.

Serban, Iulian V, Sankar, Chinnadhurai, Germain, Mathieu, Zhang, Saizheng, Lin, Zhouhan, Subramanian, Sandeep, Kim, Taesup, Pieper, Michael, Chandar, Sarath, Ke, Nan Rosemary, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

Su, Pei-Hao, Gasic, Milica, Mrksic, Nikola, Rojas-Barahona, Lina, Ultes, Stefan, Vandyke, David, Wen, Tsung-Hsien, and Young, Steve. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*, 2016.

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

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

Thrun, Sebastian and Schwartz, Anton. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

Van der Pol, Elise and Oliehoek, Frans A. Coordinated deep reinforcement learners for traffic light control. In *In proceedings of NIPS*, volume 16, 2016.

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

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Wen, Zheng, O'Neill, Daniel, and Maei, Hamid. Optimal demand response using device-based reinforcement learning. *IEEE Transactions on Smart Grid*, pp. 2312–2324, 2015.

Wu, Yonghui, Schuster, Mike, Chen, Zhifeng, Le, Quoc V, Norouzi, Mohammad, Macherey, Wolfgang, Krikun, Maxim, Cao, Yuan, Gao, Qin, Macherey, Klaus, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zahavy, Tom, Magnani, Alessandro, Krishnan, Abhinandan, and Mannor, Shie. 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)*, 2018.

Zelinka, Mikulas. Using reinforcement learning to learn how to play text-based games. *arXiv preprint arXiv:1801.01999*, 2018.

Zhao, Tiancheng and Eskenazi, Maxine. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*, 2016.

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 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 $T_{s,a}(t) \le 4\frac{\beta_t}{(u-\ell)^2} + 1$ 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 at 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:

$$x(s_t)^T \bar V_{t-1,a}^{-1} x(s_t) = x(s_t)^T \left\{ \lambda I + T_{s,a}(t-1) x(s_t) x(s_t)^T + \sum_{s' \neq s_t} T_{s',a}(t-1) x(s') x(s')^T \right\}^{-1} x(s_t)$$
$$\overset{(1)}{\le} x(s_t)^T \left\{ \lambda I + T_{s,a}(t-1) x(s_t) x(s_t)^T \right\}^{-1} x(s_t) \overset{(2)}{=}$$
$$\frac{\|x(s_t)\|^2}{\lambda} - \frac{T_{s,a}(t-1)\frac{\|x(s_t)\|^4}{\lambda^2}}{1 + T_{s,a}(t-1)\frac{\|x(s_t)\|^2}{\lambda}} = \frac{\|x(s_t)\|^2}{\lambda + T_{s,a}(t-1)\|x(s_t)\|^2} \le \frac{1}{T_{s,a}(t-1)}$$

(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. 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 achieved by placing a regular 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

$$x(s_t)^T \hat\theta_{t-1,a} - \sqrt{\beta_{t-1}(\tilde\delta) x(s_t)^T \bar V_{t-1,a}^{-1} x(s_t)} \overset{(1)}{\ge} u - 2\sqrt{\beta_{t-1}(\tilde\delta) x(s_t)^T \bar V_{t-1,a}^{-1} x(s_t)} \overset{(2)}{\ge} u - 2\sqrt{\frac{\beta_{t-1}(\tilde\delta)}{T_{s,a}(t-1)}} > \ell$$

where (1) is correct due to Equation 2 with $\mathbb{E}[e(s_t,a)] = \theta_a^{*T} x(s_t) \ge u$, with probability $1-\delta$, and (2) is correct due to Equation **??**. Therefore, if $T_{s,a}(t) \ge 4\frac{\beta_t}{(u-\ell)^2}$ 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

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

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. $\forall s, \|x(s)\|_2 \le L$, then, using the simpler form of β_t, the bound can be written as $\lim_{t\to\infty} \frac{T_{s,a}(t)}{\log\left(\frac{t}{\delta}\right)} \le \frac{4R^2 d}{(u-\ell)^2}$, which means an invalid action is sampled a logarithmic number of times.

□

B. Maps of Zork

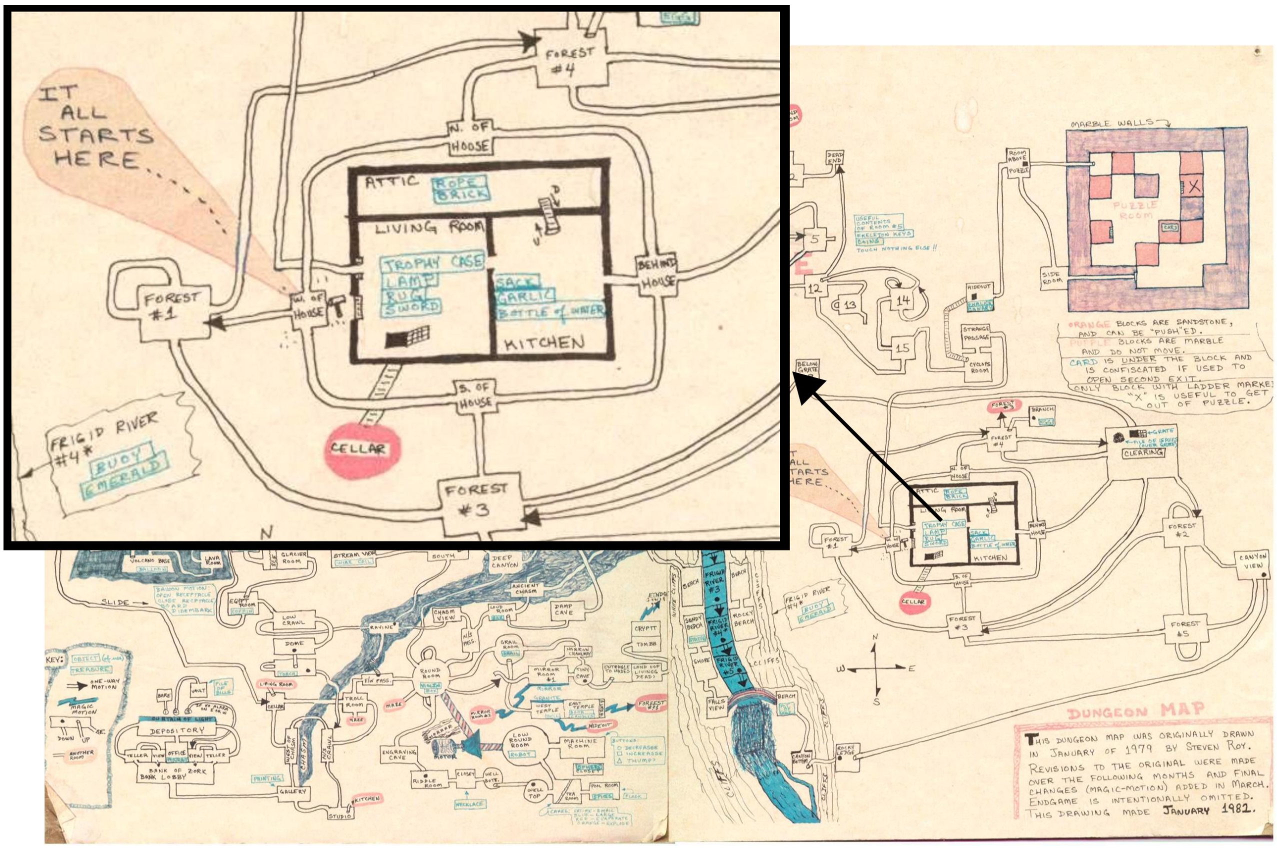


Figure 5. The world of Zork

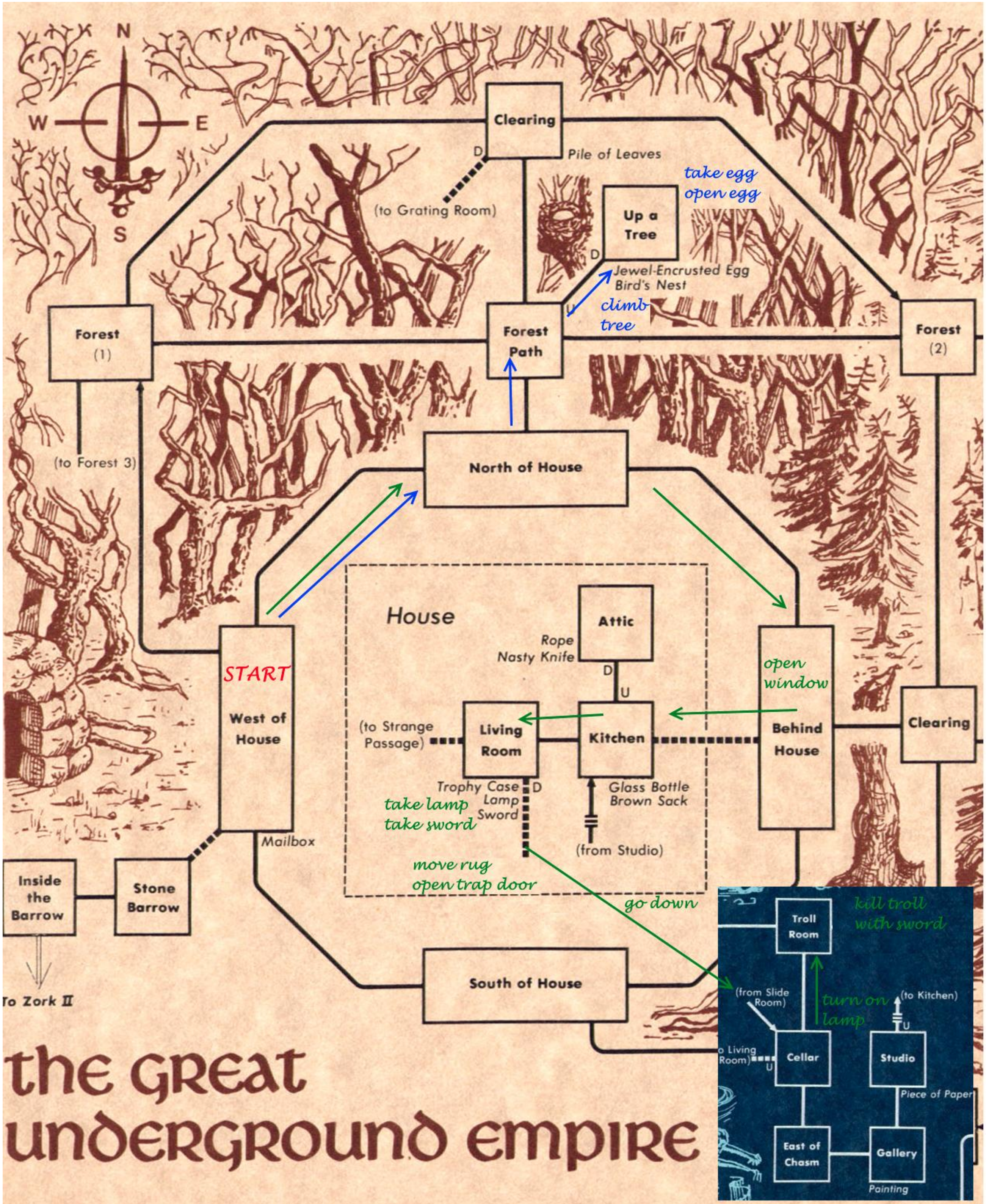


Figure 6. Subdomains of Zork; the Troll (green) and Egg (blue) Quests. Credit: S. Meretzky, The Strong National Museum of Play.