COMBATING DEEP REINFORCEMENT LEARNING'S SISYPHEAN CURSE WITH INTRINSIC FEAR

Zachary C. Lipton *
Department of Computer Science and Engineering
University of California, San Diego
zlipton@cs.ucsd.edu

Jianfeng Gao, Lihong Li, Jianshu Chen, Li Deng
Microsoft Research
Redmond, Washington
{jfgao, lihongli, jianshuc, deng}@microsoft.com

ABSTRACT

To use deep reinforcement learning in the wild, we might hope for an agent that can avoid catastrophic mistakes. Unfortunately, even in simple environments, the popular deep Q-network (DQN) algorithm is doomed by a Sisyphean curse. Owing to the use of function approximation, these agents eventually forget experiences as they become exceedingly unlikely under a new policy. Consequently, for as long as they continue to train, DQNs may periodically relive catastrophic mistakes. Many real-world environments where people might be injured exhibit a special structure. We know a priori that catastrophes are not only bad, but that agents need not ever get near to a catastrophe state. In this paper, we exploit this structure to learn a *reward shaping* that accelerates learning and guards oscillating policies against repeated catastrophes. First, we demonstrate unacceptable performance of DQNs on two toy problems. We then introduce *intrinsic fear*, a new method that mitigates these problems by avoiding dangerous states. Our approach incorporates a second model trained via supervised learning to predict the probability of catastrophe within a short number of steps. This score then acts to penalize the Q-learning objective, shaping the reward function away from catastrophic states.

1 INTRODUCTION

Following success on Atari games (Mnih et al., 2015) and the board game Go (Silver et al., 2016), many researchers have begun exploring practical applications of deep reinforcement learning (DRL). With DRL, we might hope to replace simple hand-coded control policies with continuously learning agents. Some investigated applications include robotics (Levine et al., 2016), dialogue systems (Fatemi et al., 2016; Lipton et al., 2016), energy management (Night, 2016), and self-driving cars (Shalev-Shwartz et al., 2016). Amid this push to apply DRL, we might ask, *can we trust these agents in the wild?*

Agents acting in real world environments might possess the ability to cause catastrophic outcomes. Consider a self-driving car that might hit pedestrians, or a domestic robot that might injure a child. We might hope to prevent DRL agents from ever making catastrophic mistakes. But doing so requires extensive prior knowledge of the environment in order to constrain the exploration of policy space (Garcıa & Fernández, 2015). Such knowledge may not always exist. In this paper, we don't assume prior knowledge of the state-space.

Already, we should note that multiple, conflicting definitions of safety and catastrophe exist, a problem that invites further philosophical consideration. We will return to this matter in Section 3. For now, we introduce a specific but plausible notion of a catastrophe. Suppose that catastrophes are a subset of states that are terminal, yielding a return of 0 and that returns are greater than or equal to

*http://zacklipton.com

0 for non-catastrophic states. Moreover, we assume that an optimal policy never even comes *near* a catastrophe state. Note that this doesn't hold in general RL problems. In a cliff world, the pot of gold might lie on the ledge of a cliff. But for a self-driving car, we don't suspect that it ever should have to come within a split-second decision of committing murder. Finally, while we don't assume prior knowledge of which states are dangerous, we do assume the existence of catastrophe detector. After encountering a catastrophic state, an agent can realize this and take action to avoid dangerous states in the future.

Given this definition, we address two challenges: First, can we expect DRL agents, after experiencing some number of catastrophic failures, to avoid perpetually making the same mistakes? Second, can we use our prior knowledge that catastrophes should be kept at distance to accelerate learning of a DRL agent? Our experiments show that even on toy problems, the deep Q-network (DQN), a basic algorithm behind many of today's state of the art DRL systems, struggles on both counts. Even in toy environments, DQNs may encounter thousands of catastrophes before avoiding them and may periodically repeat old errors so long as they continue to learn. We call this latter problem the Sisyphean curse.

This poses a formidable obstacle to using DQNs in the real world. How can we hand over responsibility for any consequential actions (control of a car, say) to a DRL agent if it may be doomed to periodically remake every kind of mistake, however grave, so long as it continues to learn? Imagine a self-driving car that had to periodically hit a few pedestrians in order to remember that it's undesirable.

Note that traditional tabular agents do not suffer from the Sisyphean curse. In the tabular setting, an RL agent never forgets the learned dynamics of its environment, even as its policy evolves. Thus, if the Markovian assumption holds, eventual convergence to a globally optimal policy is guaranteed. Unfortunately, the tabular approach becomes infeasible in high-dimensional, continuous state spaces.

The trouble owes to the use of function approximation (Murata & Ozawa, 2005). When training a DQN, we successively update a neural network based on experiences. These experiences might be sampled in an online fashion, from a trailing window (*experience replay buffer*), or uniformly from all past experiences. Regardless of which mode we use to train the network, eventually, states that a learned policy never encounters will come to form an infinitesimally small region of the training distribution. At such time, our networks are subject to the classic problem of catastrophic interference McCloskey & Cohen (1989); McClelland et al. (1995). Nothing prevents the DQN's policy from drifting back towards one that revisits catastrophic, but long-forgotten mistakes.

More formally, we could characterize the failures as:

- Training under distribution $\mathcal{D}$, our agent produces a safe policy π_s that avoids catastrophes
- Collecting data generated under π_s yields a distribution $\mathcal{D}'$
- Training under $\mathcal{D}'$, the agent produces π_d, a policy that once again experiences catastrophes

In this paper, we illustrate the brittleness of modern deep reinforcement learning algorithms. We introduce a simple pathological problem called *Adventure Seeker*. This problem consists of a one-dimensional continuous state, two actions, simple dynamics and a clear analytic solution. Nevertheless, the DQN fails. We then show that similar dynamics exist in the classic RL environment Cart-Pole. Finally, we present preliminary findings on the Atari game Seaquest, suggesting that the intrinsic fear model may also be useful for improving performance in more complex environments.

To combat these problems, we propose *intrinsic fear* (Figure 1). In this heuristic approach, alongside the DQN, we train a supervised *danger model*. The danger model predicts which states are likely to lead to a catastrophe within some number of steps k_r. The output of this model (a probability) is then scaled by a *fear factor* and used to penalize the Q-learning target. Our approach bears some resemblance to intrinsic motivation Chentanez et al. (2004). But instead of perturbing the reward function to encourage the discovery of novel states, we perturb it to discourage the rediscovery of catastrophic states.

2 Background: Deep Q-Learning

For context we briefly review deep Q-learning. Over a series of turns, an agent interacts with its environment via a Markov decision process, or MDP, $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. At each step t, an agent

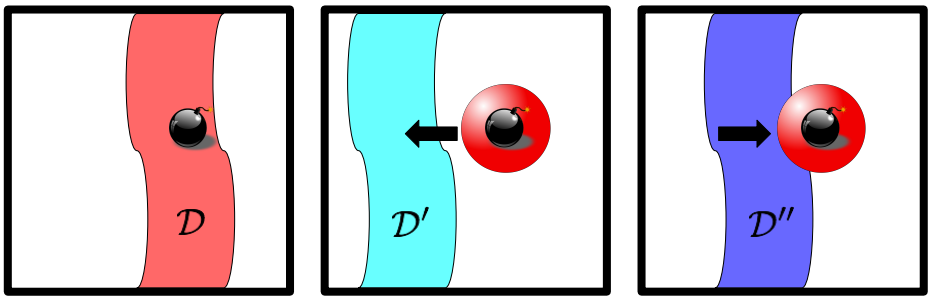


Figure 1: With typical deep reinforcement learning techniques, an agent may forget about catastrophic failure modes as they become unlikely under an updated policy. With *intrinsic fear*, we learn to recognize danger zones (red circles) around catastrophes and shape reward functions away from these zones.

observes a state $s \in \mathcal{S}$. The agent then chooses an action $a \in \mathcal{A}$ according to some policy π. In turn, the environment transitions to a new state $s_{t+1} \in \mathcal{S}$ according to transition dynamics $\mathcal{T}(s_{t+1}|s_t, a_t)$ and generates a reward r_t with expectation $\mathcal{R}(s, a)$.. This cycle continues until each episode terminates.

The goal of an agent is to maximize the cumulative discounted return $\sum_{t=0}^{T} \gamma^t r_t$. Temporal-differences (TD) methods (Sutton, 1988) such as Q-learning (Watkins & Dayan, 1992) model the Q-function, which gives the *optimal* discounted total reward of a state–action pair; the greedy policy w.r.t. the Q-function is optimal (Sutton & Barto, 1998). Most problems of practical interests have large state spaces, thus the Q-function has to be approximated by parametric models such as neural networks.

In deep Q-learning, this is typically accomplished by alternately collecting experiences by acting greedily with respect to $Q(s, a; \theta_Q)$ and updating the parameters θ_Q. Updates proceed as follows. For a given experiences (s_t, a_t, r_t, s_{t+1}), we minimize the squared Bellman error:

$$\mathcal{L} = (Q(s_t, a_t; \theta_Q) - y_t)^2 \tag{1}$$

for $y_t = r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'; \theta_Q)$. Traditionally, the parameterised $Q(s, a; \theta)$ is trained by stochastic approximation, estimating the loss on each experience as it is encountered, yielding the update:

$$\theta_{t+1} \leftarrow \theta_t + \alpha(y_t - Q(s_t, a_t; \theta_t))\nabla Q(s_t, a_t; \theta_t). \tag{2}$$

Q-learning methods also require an exploration strategy, and for simplicity we consider only the ϵ-greedy heuristic. However, our techniques apply equally for Thompson-sampling based exploration. For a thorough overview of RL fundamentals, we refer the reader to Sutton & Barto (1998).

A few tricks are often useful to stabilize Q-learning with function approximation. Of particular relevance to this work is experience replay (Lin, 1992): the RL agent maintains a buffer of past experiences, applying TD-learning on randomly selected mini-batches of experience to update the Q-function. This technique has proven effective to make Q-learning more stable and more data-efficient (Lin, 1992; Mnih et al., 2015).

3 Safe Reinforcement Learning

At the outset we ought to consider a more formal treatment of safe reinforcement learning. Precisely what do we mean by *safety*, *catastrophe*, or *danger*? The literature offers several notions of safety in reinforcement learning. Garcıa & Fernández (2015) provide a comprehensive review of this literature, suggesting that existing approaches can be divided into two categories.

The first approach consists of modifying the objective function. Traditionally, in a Markov decision process, a reinforcement learner seeks to maximize expected return. Hans et al. (2008) defines a *fatality* as a return below some threshold τ. In this view, an agent might seek to minimize the worst

case scenario instead of maximizing the expected return. Along these lines, Heger (1994) suggested the $\hat{Q}$-learning objective:

$$\hat{Q} = \min(\hat{Q}(s_t, a_t), r_{t+1} + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{Q}(s_{t+1}, a_{t+1}))$$

Other relevant papers suggest modifying the objective to address risk. While classic Q-learning objective addresses only expected returns, alternative objectives penalize policies for returns with high-variance Garcıa & Fernández (2015). Other suggest incorporating external knowledge into the exploration process. In the extreme case, we might confine our policy search to the subset of policies known to be *safe*.

We see the following problem. The first category of methods acts only on the summary statistics of the return. This can be beneficial because it requires no foreknowledge. But the drawback is that they fundamentally change the objective everywhere, even in states that we might not be worried about. On the other hand, processes relying on expert knowledge may presume an unreasonable level of foreknowledge.

In this paper, we propose a new formulation of the safety problem. We suppose there exists a subset $\mathcal{C} \subset \mathcal{S}$ of catastrophic states. These states have minimum return, and thus resemble the fatalities of Hans et al. (2008). Additionally we assume that it's reasonable for policy to avoid even getting close to these states.

In other words, there exist good policies that never get within a few hops of a catastrophe state. Just as a child shouldn't run with scissors, a self-driving car should never be one split-second decision away from killing a passenger or pedestrian. Note, this imposes some prior knowledge. In a general, unknown environment, it might be that a good policy should get very close to a fatality in order to maximize reward.

We don't assume advanced knowledge of C, but we do assume the agent possesses a *catastrophe detector*. In the case of a self-driving car, this might be an impact detector. Even if we can't anticipate all the situations which might lead to an accident, we can recognize one when it has already happened and take measures to avoid danger zones around these states in the future.

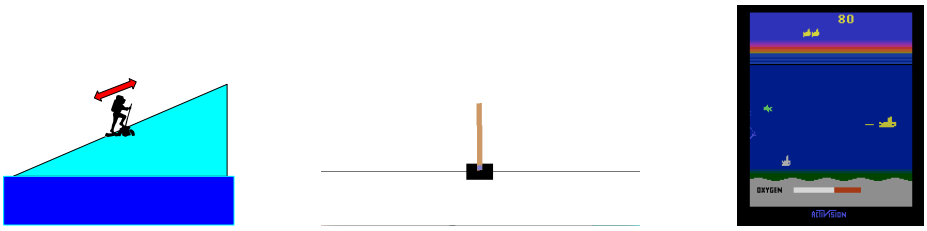
The notion of danger zone suggests some measure of proximity. Informally, we conceive of *danger* as the likelihood of entering a catastrophe state $c \in \mathcal{C}$ within a short number of steps. Note, this probability depends on the policy. In this paper, we collect the experiences used to train the danger model from the entire training period. So the likelihood here is determined based on the observations of many policies over the course of training. Explicitly modeling danger zones can be useful for two reasons. First, it provides a mechanism to avoid oscillating policy learners from repeatedly encountering catastrophe states. Second, reward-shaping to avoid danger zones can significantly reduce the sample complexity of exploration.

4 Two Pathological Failure Cases

We now present two pathological environments. In the *Adventure Seeker* (Figure 2a) environment, we imagine a player placed on a hill, sloping upward to the right. At each turn, the player can move to the right (up the hill) or left (down the hill). The environment adjusts the player's position accordingly, adding some random noise. Between the left and right edges of the hill, the player gets more reward for spending time higher on the hill. But if the player goes too far to the right, she will fall off (a *catrastrophic state*), terminating the episode and receiving a return of 0.

Formally, the state consists of a single continuous variable $s \in [0, 1.0]$, denoting the player's position. The starting position for each episode is chosen uniformly at random in the interval $[.25, .75]$. The available actions consist only of $\{-1, +1\}$ (*left* and *right*). Given an action a_t in state s_t, $\mathcal{T}(s_{t+1}|s_t, a_t)$ gives successor state $s_{t+1} \leftarrow s_t + .01 \cdot a_t + \eta$ where $\eta \sim \mathcal{N}(0, .01^2)$. At each turn, the player gets reward equal to s_t (proportional to height). The player falls off the hill, entering the catastrophic terminating state, whenever an action would result in successor state $s_{t+1} > 1.0$ or $s_{t+1} < 0.0$.

This game admits an obvious analytic solution. There exists some threshold above which the agent should always choose to go left, and below which it should always go right. Even an infant could grasp the gist of this solution. And yet a state-of-the-art DQN model learning online or with experience



(a) Adventure Seeker (b) Cart-Pole (c) Seaquest

Figure 2: In experiments, we consider two toy environments (a,b) and the Atari game Seaquest (c)

replay successively plunges to its death. To be clear, the DQN does learn a near-optimal thresholding policy quickly. But over the course of continued training, the agent oscillates between a reasonable thresholding policy and one which always moves right, regardless of the state. The pace of this oscillation evens out and all networks (over multiple runs) quickly reach a constant catastrophe per turn rate (Figure 3a) that does not attenuate with continued training. How could we ever trust a system that can't solve *Adventure Seeker* to make consequential real-world decisions?

Cart-Pole (Figure 2b) is a classic RL environment in which an agent tries to balance a pole atop a cart. Qualitatively, the game exhibits four distinct catastrophe modes. The pole could fall down to the right or fall down to the left. Additionally, the cart could run off the right boundary of the screen or run off the left.

Formally, at each time, the agent observes a four-dimensional state vector (x, v, θ, ω) consisting respectively of the cart position, cart velocity, pole angle, and the pole's angular velocity. At each time step, the agent chooses an action, applying a force of either -1 or $+1$. For every time step that the pole remains upright and the cart remains on the screen, the agent receives a reward of 1. If the pole falls, the episode terminates, giving a return of 0 from the penultimate state. In experiments, we use the implementation *CartPole-v0* contained in the openAI gym Brockman et al. (2016). Like Adventure Seeker, this problem admits an analytic solution. A perfect policy should never drop the pole. But, as with Adventure Seeker, a DQN converges to a constant rate of catastrophes per turn.

In addition to these pathological cases, we also present some preliminary experiments extending these results to the Atari game environment. We consider Seaquest, a game in which a player swims under water. The player can navigate up, down, left, right, and shoot a gun. Periodically, as the oxygen gets low, she must rise to the surface for oxygen. Additionally, fish swim across the screen. The player gains points each time she shoots a fish. Colliding with a fish or running out of oxygen result in death. The player has 3 lives, and the final death is a terminal state. In this paper, because the environment doesn't explicitly report intermediate deaths, we consider only the final death to be the catastrophe state. In the future work, we might hack the game to include all deaths.

5 Intrinsic Fear

We now introduce intrinsic fear (Algorithm 1), a novel mechanism for avoiding catastrophes when learning online with function approximation. In our approach, we maintain both a DQN and a separate, supervised *danger model*. Our *danger model* provides an auxiliary source of reward, penalizing the Q-learner for entering dangerous states.

The goal in learning this smooth danger zone is twofold. First the knowledge that regions around certain states should be avoided represents a source of prior knowledge that can accelerate learning in some environments. Second, by using danger zones to shape rewards away from catastrophic states, we allow oscillating policies to drift close (but not too close) to catastrophe states, giving them opportunity to adjust trajectories away, without having to re-experience the catastrophe. We draw some inspiration from the idea of a parent scolding a child for running around with a knife. The child can learn to adjust its behavior without actually having to stab someone.

We make the following conservative assumption. While we don't know anything in advance about the state space, we possess a *catastrophe detector* which can identify each catastrophe as it happens.

We also assume that the agent collects a buffer of previously observed states. After it detects each catastrophe, the agent can refer back to the trajectory that led up to it.

Algorithm 1: Training DQN with Intrinsic Fear

function Train($Q(\cdot,\cdot;\cdot)$, $d(\cdot;\cdot)$, T, λ, k_λ, k_r);
Input : Two model architectures: Q (a DQN) and d (the danger model), time limit T, fear factor λ, fear phase-in length k_λ, fear radius k_r
Output : Learned parameters θ_Q and θ_d
Initialize parameters θ_Q and θ_d randomly
Initialize replay memory $\mathcal{D}_e$, danger states $\mathcal{D}_d$, and safe states $\mathcal{D}_s$
for *t in 1:T* **do**
  With probability ϵ select random action a_t
  Otherwise, select action $a_t = \max_{a'} Q(s_t, a'; \theta_Q)$
  Execute action a_t in environment, observing reward r_t and successor state s_{t+1}
  Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
  if *s_{t+1} is a terminal state* **then**
    Add states s_{t-k_r} through s_t to $\mathcal{D}_d$
    Add states s_{t-n_e} through s_{t-k_r-1} to $\mathcal{D}_s$
  end
  Sample random minibatch of transitions (s_τ , a_τ, r_τ, $s_{\tau+1}$) from $\mathcal{D}$
  Set $\lambda_t \leftarrow \min(\lambda, \frac{\lambda \cdot t}{k_\lambda})$
  Set $y_t \leftarrow \left\{ \begin{array}{ll} r_t - \lambda_t, & \text{for terminal } s_{\tau+1} \\ r_t + \max_{a'} Q(s_{t+1}, a'; \theta_Q) - \lambda \cdot d(s_{t+1}; \theta_d) & \text{for non-terminal } s_{t+1} \end{array} \right\}$
  Apply SGD step $\theta_Q \leftarrow \theta_Q - \eta \cdot \nabla_{\theta_Q}(y_\tau - Q(s_\tau, a_\tau; \theta_Q))^2$
  Sample random mini-batch s_j with 50% of examples from $\mathcal{D}_d$ and 50% from $\mathcal{D}_s$
  Set $y_j \leftarrow \left\{ \begin{array}{ll} 1, & \text{for } s_j \in \mathcal{D}_d \\ 0, & \text{for } s_j \in \mathcal{D}_s \end{array} \right\}$
  Apply SGD step $\theta_d \leftarrow \theta_d - \eta \cdot \nabla_{\theta_d} \text{loss}_d(y_j, d(s_j; \theta_d))$
end

The technique works as follows: In addition to the DQN, we maintain a binary classifier that we term a *danger model*. In our case, it is a neural network of the same architecture as the DQN (but for the output layer). In general, it could be any supervised model. The danger model's purpose is to identify the likelihood that any state will lead to catastrophe within k moves. In the course of training, our agent adds each experience (s, a, r, s') to the experience replay buffer. As each catastrophe is reached at the n_{th} turn of an episode, we add the k_r (*fear radius*) states leading up to the catastrophe to a list of *danger states*. We add the preceding $n - k_r$ states to a list of *safe states*. When $n < k_r$, all states for that episode are added to the list of danger states.

Then after each turn, in addition to making one update to the Q-network, we make one mini-batch update to the danger model. To make this update, we sample 50% of states from the *danger states*, assigning them label 1 and 50% of states from the *safe states*, assigning them label 0.

For each update to the DQN, we perturb the TD target y_t. Instead of updating $Q(s_t, a_t; \theta_Q)$ towards $r_t + \max_{a'} Q(s_{t+1}, a'; \theta_Q)$, we introduce the *intrinsic fear* to the model via the target:

$$y_t^{IF} = r_t + \max_{a'} Q(s_{t+1}, a'; \theta_Q) - \lambda \cdot d(s_{t+1}; \theta_d)$$

where $d(s; \theta_d)$ is the danger model and λ is a *fear factor* determining the scale of the impact of intrinsic fear on the Q update.

Note that our method perturbs the objective function. As such, one might be reasonably concerned that it might imply a different optimal policy. We can address this concern in the following way. Let us assume the optimal policy never enters a danger zone of radius k_r. Let us also assume that over course of training, the danger model approaches perfect accuracy, meaning that all safe states s are given danger score $d(s) = 0$. Then because our penalty is equal to 0 for all safe states and $>= 0$ for all danger states, the optimal policy receives the same return as under the original reward function. However, the subset of suboptimal policies that enter danger zones receive strictly worse returns.

Therefore, as the danger model approaches perfect performance wrt false positives, the optimal policy should be unchanged.

We also note that the danger zone depends on the policy. We could determine its size objectively by reference to the worst possible policy. In practice, however, we have found that we achieve better results by setting a wider fear radius than this worst-case analysis suggests.

Finally, we observe that the setting of the fear radius encodes prior knowledge of a problem. However, this prior knowledge is expressed far more succinctly. It's not obvious how we could communicate information about the high-dimensional, continuous state space with similar efficiency.

Over the course of our experiments, we discovered the following pattern. To be effective, the *fear radius* k_r should be large enough that the model can detect danger states at a safe distance where they can still be avoided. When the fear radius is too small, the danger probability is only nonzero at a points from which catastrophes are inevitable and intrinsic fear doesn't help. On the other hand, when the *fear radius* is too large, all states are predicted to be danger states. For example, early in training, all Cart-Pole experiences are shorter than 20 experiences. So a *fear radius* of 20 leads our models model to diverge. To overcome this problem, we gradually phase in the *fear factor* from 0 to λ reaching full strength at predetermined time step k_λ. In our Cart-Pole experiments we phase in over $1M$ steps.

6 EXPERIMENTS

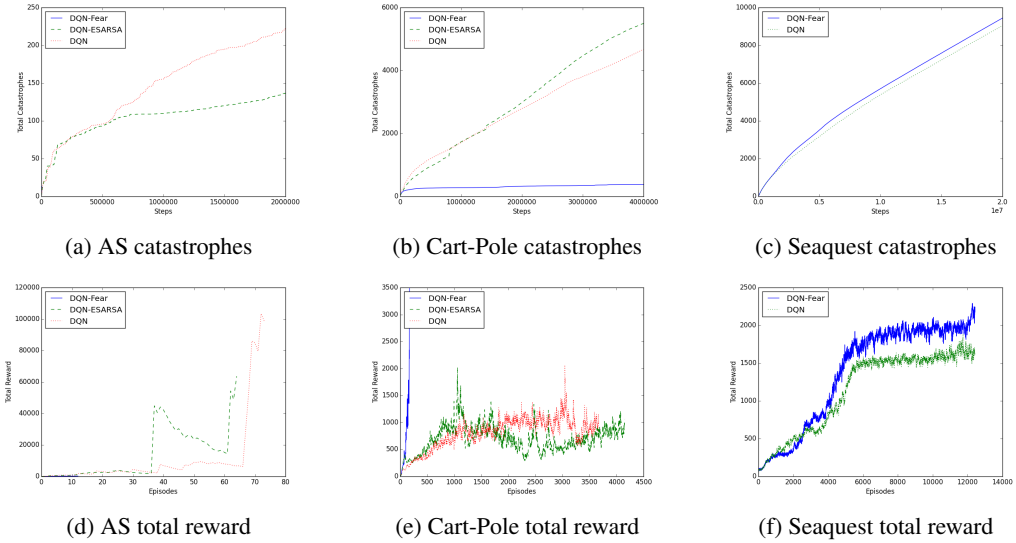
To assess the effectiveness of the intrinsic fear model, we evaluate both a standard DQN (DQN-NoFear) and one enhanced by *intrinsic fear* (DQN-Fear). In both cases, we use multilayer perceptrons (MLPs) with a single hidden layer and 128 hidden nodes. We train all MLPs by stochastic gradient descent using the Adam optimizer Kingma & Ba (2015) to adaptively tune the learning rate. Some might wonder whether the problems we observe truly owe to distributional shift or if they actually stem from the well-known problems of off-policy learning. To show that learning on-policy learning does not mitigate these issues, we also present results for an expected-SARSA variant of the DQN for Adventure Seeker and Cart-Pole.

Because, for the *Adventure Seeker* problem, an agent can escape from danger with only a few time steps of notice, we set the fear radius k_r to 5. We phase in the fear factor quickly, reaching full strength in just 1000 moves. On this problem we set the fear factor λ to 40.

For *Cart-Pole*, we set a wider fear radius of $k_r = 20$. We initially tried training this model with a shorter fear radius but made the following observation. Some models would learn well surviving for millions of experiences, with just a few hundred catastrophes. This compared to a DQN (Figure 3) which would typically suffer 4000-5000 catastrophes. When examining the output from the danger models on successful vs unsuccessful runs, we noticed that the unsuccessful models would output danger of probability greater than $.5$ for precisely the 5 moves before a catastrophe. But by that time it would be too late for an agent to correct course. In contrast, on the more successful runs, the danger model would often output predictions in the range $.1 - .5$. We suspect that this gradation between mildly dangerous states and those where danger is imminent provided a richer reward signal to the DQN. Accordingly, we lengthened the fear radius to 20 so that some states might be ambiguous (could show up in either *danger list* $\mathcal{D}_d$ or *safe list* $\mathcal{D}_s$.

On both the Adventure Seeker and Cart-Pole environments, the DQNs augments by intrinsic fear far outperform their otherwise identical counterparts (Figure 3). We compared this approach against some traditional approaches, like memory based methods for preferentially sampling failure cases but they could not improve over the DQN.

On the Atari game Seaquest, we use a fear radius of 10 and a fear factor of $.1$. Interestingly, the models trained with Intrinsic Fear have indistinguishable catastrophe rates but achieve significantly higher reward. Using a much higher fear factor and wider radius we are able to significantly lower the catastrophe rate but this improvement comes at the expense of total reward. These results suggest an interesting interplay between the various reward signals that warrants further exploration.



(a) AS catastrophes (b) Cart-Pole catastrophes (c) Seaquest catastrophes

(d) AS total reward (e) Cart-Pole total reward (f) Seaquest total reward

Figure 3: Total catastrophes and total reward for DQNs, an ESARSA variant, and *Intrinsic Fear* DQNs on Adventure Seeker, Cart-Pole and the Atari game Seaquest. All results on AS and Cart-Pole averaged over 5 runs, Atari results averaged over 3 runs. On Adventure Seeker, all Intrinsic Fear models achieve immortality within 14 runs, giving unbounded total reward and 0 catastrophes thereafter. On Seaquest, for fear factor setting of 10, the IF model achieves a near-identical catastrophe rate but significantly higher total reward.

7 Related Work

The paper addresses safety in RL, intrinsically motivated RL, and the stability of Q-learning with function approximation under distributional shift. Our work also has some connection reward shaping. While space constraints preclude adequate treatment of all prior work, we attempt to highlight the most relevant papers here.

The Stability of RL with Function Approximation: The potential oscillatory or divergent behavior of Q-learners with function approximation has been previously identified (Boyan & Moore, 1995; Baird et al., 1995; Gordon, 1996). But these papers address neither AI safety nor RL. Murata & Ozawa (2005) addresses the problem of catastrophic forgetting owing to distributional shift in RL with function approximation, proposing a memory-based solution. Outside of reinforcement learning, a number of papers address problems related to non-stationary data, including a book on covariate shift (Sugiyama & Kawanabe, 2012).

Safety in Reinforcement Learning: Several papers address safety in RL. Garcıa & Fernández (2015) provide a thorough review on the topic, dividing the existing works into those which perturb the objective function and those which use external knowledge to improve the safety of exploration. Hans et al. (2008) defines a fatal state as one from which one receives a return below some safety threshold τ. They propose a solution comprised of two components. One, the *safety function*, identifies unsafe states. The other, denoted the *backup model*, is responsible for navigating away from the critical state. Their work does not address function approximation, instead focusing on a domain where a tabular approach is viable. Moldovan & Abbeel (2012) gives a definition of safety based on ergodicity. They consider a fatality to be a state from which one cannot return to the start state. Shalev-Shwartz et al. (2016) provides theoretical analysis considering how strong a penalty should be to discourage accidents. They also consider hard constraints to ensure safety, an approach that differs from ours. None of the above works address the case where distributional shift dooms an agent to perpetually revisit catastrophic failure modes.

Intrinsically Motivated Reinforcement Learning: A number of papers have investigated the idea of intrinsically motivated reinforcement learners. Intrinsic motivation refers to an intrinsically assigned reward, in contrast to the extrinsic reward that comes from the environment. Typically

intrinsic motivation is proposed as a way to encourage exploration of an environment (Schmidhuber, 1991; Bellemare et al., 2016) and to acquire a modular set of skills Chentanez et al. (2004). In principle, such motivation can lead agents to explore intelligently even when extrinsic rewards are sparse. Some papers refer to the intrinsic reward for discovery as *curiosity*. Like classic work on intrinsic motivation, our methods operate by perturbing the reward function. But instead of assigning bonuses to encourage discovery of novel transitions, we assign penalties to discourage re-discovery of catastrophic transitions.

8 DISCUSSION

Our experiments suggest that DQNs may be too brittle for use in real-world applications where harm can come of actions. While it's easy to visualize these problems on toy examples, similar dynamics are embedded in more complex domains. Consider a domestic robot acting as a barber. The robot might receive positive feedback for giving a closer shave. This reward encourages closer contact at a steeper angle. Of course, the shape of this reward function belies the catastrophe lurking just past the optimal shave. Similar dynamics might be found in a vehicle which is rewarded for traveling faster but could risk an accident with excessive speed.

These scenarios are not so unlike the pathological case presented in Adventure Seeker. We might even say these shortcomings seem analogous to the fundamental inability of linear models to solve XOR. But while XOR can be solved by simply incorporating hidden layers of representation, it's less obvious how to overcome these pitfalls.

These early successes of the intrinsic fear model suggest that for some classes of problems, we might avoid perpetual catastrophe and still enjoy the benefits of function approximation. In this work we assume the ability to recognize a catastrophe once it has happened. But we don't assume anything in advance about the precise form that the danger states might take in the state space. Our approaches seem appropriate for problems with some notion of proximity. A self-driving car can't run over a passenger in the next second if no people are in close proximity. But out methods might not be appropriate for other problems. For example, in a problem where a single action, from any state, can produce a catastrophe with high probability, our methods might fail.

This work represents a first step towards combating AI safety issues stemming from the use of function approximation in deep reinforcement learning. In follow-up work, we hope to formalize the notion of danger presented here. We hope to critically examine competing definitions of catastrophe and of danger, to organize them into a taxonomy and to develop theory addressing the most promising ones. We also hope to explore the effectiveness of our technique on more complex domains. In precisely what sorts of environments should our approach work? Under what conditions can we expect it to fail?

REFERENCES

Leemon Baird et al. Residual algorithms: Reinforcement learning with function approximation. 1995.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.

Justin Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. In *NIPS*, 1995.

Greg Brockman et al. OpenAI gym. *arXiv:1606.03152*, 2016.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *NIPS*, 2004.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. Policy networks with two-stage training for dialogue systems. In *SIGDIAL*, 2016.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 2015.

Geoffrey J Gordon. Chattering in SARSA(λ) - a CMU learning lab internal report. 1996.

Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. Safe exploration for reinforcement learning. In *ESANN*, 2008.

Matthias Heger. Consideration of risk in reinforcement learning. In *Machine Learning*, 1994.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Sergey Levine et al. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 1992.

Zachary C Lipton et al. Efficient exploration for dialogue policy learning with BBQ networks & replay buffer spiking. In *NIPS Workshop on Deep Reinforcement Learning*, 2016.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 1995.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 1989.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes. In *ICML*, 2012.

Makoto Murata and Seiichi Ozawa. A memory-based reinforcement learning model utilizing macro-actions. In *Adaptive and Natural Computing Algorithms*. Springer, 2005.

Will Night. The AI that cut googles energy bill could soon help you. *MIT Tech Review*, 2016.

Jurgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior (SAB90)*. Citeseer, 1991.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv:1610.03295*, 2016.

David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Masashi Sugiyama and Motoaki Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT Press, 2012.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

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

Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.