# 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 would never make catastrophic mistakes. At the very least, we could hope that an agent would eventually learn to avoid old mistakes. Unfortunately, even in simple environments, modern deep reinforcement learning techniques are 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, state-aggregating agents may periodically relive catastrophic mistakes. We demonstrate unacceptable performance of deep Q-networks on two toy problems. We then introduce *intrinsic fear*, a method that mitigates these problems by avoiding dangerous 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). The investigated applications include robotics (Levine, 2016), dialogue systems (Fatemi et al., 2016; Lipton et al., 2016), energy management (Night, 2016), and even self-driving cars (Shalev-Shwartz et al., 2016). Amid this push to commercialize 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. Can we ensure that DRL agents will never make catastrophic mistakes? In this paper, we address a more modest challenge: can we reasonably expect DRL agents, after experiencing some number of catastrophic failures, to avoid perpetually making the same mistakes?

In the tabular setting, with a finite state space, an RL agent never forgets the learned dynamics of the environment, even as its policy evolves. Thus, eventual convergence to a globally optimal policy is guaranteed. However, the tabular approach becomes infeasible as state spaces grow large.

Even on toy problems, today's state of the art deep reinforcement learning algorithms may perpetually repeat old mistakes. The trouble owes to the use of function approximation. With deep reinforcement learning, 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. Note, for conventional feedforward neural networks, these problems owing to distributional shift do not depend on the choice of architecture. Thus both agents equipped with deep nets and those using linear approximations to Q suffer similarly.

*http://zacklipton.com

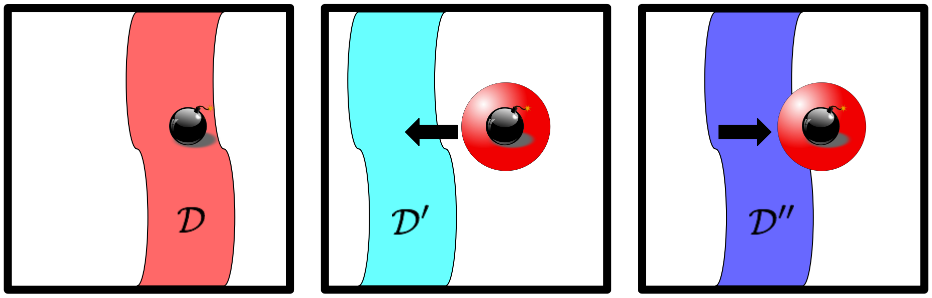



Figure 1: With typical deep reinforcement learning techniques, an agent may forget catastrophic failure modes, however simple, as they become unlikely under an updated policy. With *intrinsic fear*, we learn to recognize danger zones (red circles) around catastrophes. We then shape reward functions away from these zones, guarding oscillating policies against making old mistakes.

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; McClelland et al. (1995). Nothing prevents our policies from drifting back towards repeating disastrous, but long-forgotten mistakes.

More formally, we could characterize the source of failure as:

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

This poses a tremendous obstacle to ever using modern DRL 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?

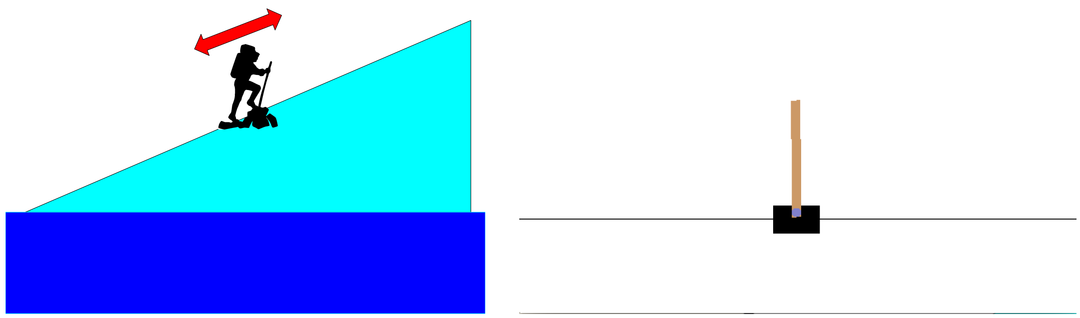
In this paper, we illustrate the remarkable brittleness of modern deep reinforcement learning algorithms. We introduce a simple pathological problem called *Adventure Seeker*. On this problem, with one-dimensional continuous state, two actions, and a clear analytic solution, the DQN fails. We then show that similar dynamics exist in the classic Cart-Pole environment.

To combat these problems, we propose *intrinsic fear* (Figure 1). In this approach, alongside the DQN, we train a supervised *danger model*. This model predicts which states are likely to lead to a catastrophe within some number of steps. The output of this model (a probability) is then scaled by a *fear factor* and used to perturb the reward function. Our approach bears some resemblance to intrinsic motivation Chentanez et al. (2004). However, 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

We briefly review deep Q-learning. Over a series of turns, an agent interacts with its environment via a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. At each step $t$, an agent observes a state $s \in \mathcal{S}$. The agent then chooses an action $a \in \mathcal{A}$ according to some policy $\pi$. The environment then 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) learn 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.



(a) Adventure Seeker (b) Cart-Pole

Figure 2: Two simple pathological environments

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

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

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

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

Q-learning strategies also require an exploration strategy, and for simplicity we consider only the $\epsilon$-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 Two Pathological Failure Cases

Given the success of DQNs on complex domains, absent deep knowledge of RL, it might seem surprising that they fail badly on toy environments. We present two such pathological cases here.

### 3.1 Adventure Seeker

To present a simple failure case, we introduce the *Adventure Seeker* (Figure 2a) environment. Here, 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, 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, terminating the episode, 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 replay successively plunges to its death. To be clear, the agent does in fact recover a near-optimal thresholding policy quickly. But over the course of continued training, the agent oscillates between a 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?

### 3.2 CART-POLE

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 failure modes. The pole could fall down to the right or fall down to the left. Also, 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 $(x, v, \theta, \omega)$ 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.

## 4 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*. By danger, we mean that a state might, within a short number of steps and with substantial probability, lead to a catastrophic failure. Our *danger model* provides an auxiliary source of reward, penalizing the Q-learner for entering dangerous states.

Our goal is to allow a policy to drift close (but not too close) to catastrophe states, giving it 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 must be able to identify each catastrophe as it happens. Implicitly, we assume that a reasonable policy could avoid ever being in close temporal proximity to a potential catastrophe. For the case of Adventure Seeker and Cart-Pole, a catastrophe is simply the terminal state.

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 conceivably, it could be any supervised model, perhaps sharing representations with the DQN. 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.

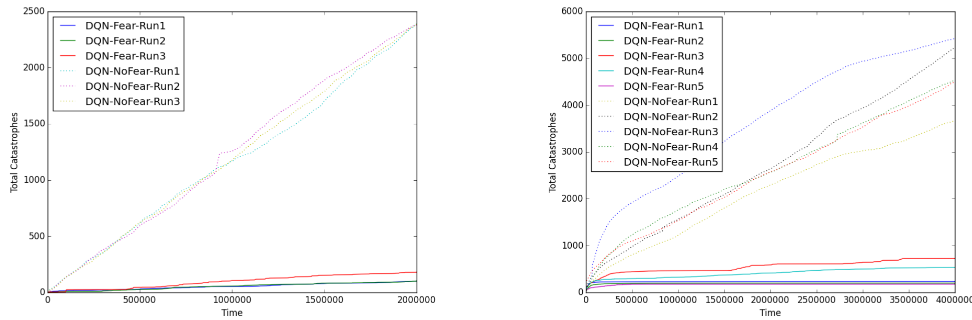
Then after each turn, in addition to making one update to the Q-network, we make one mini-batch update to the danger model. To make this update, we sample $50\%$ of states from the list *danger states*, assigning them label 1 and $50\%$ of states from the *safe states*, assigning them label 0.

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:

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

**Algorithm 1:** Training DQN with Intrinsic Fear

function Train( $Q(\cdot,\cdot;\cdot)$, $d(\cdot;\cdot)$, $T$, $\lambda$, $k_\lambda$, $k_r$);
**Input** : Two model architectures: $Q$ (a DQN) and $d$ (the danger model), time limit $T$, fear factor $\lambda$, fear phase-in length $k_\lambda$, fear radius $k_r$
**Output** : Learned parameters $\theta_Q$ and $\theta_d$
Initialize replay memory $\mathcal{D}_e$ to empty list $\emptyset$
Initialize danger states $\mathcal{D}_d$ to empty list $\emptyset$
Initialize safe states $\mathcal{D}_s$ to empty list $\emptyset$
Initialize DQN parameters $\theta_Q \sim \mathcal{N}(0, 0.01)^{dim(\theta_Q)}$
Initialize danger model parameters $\theta_d \sim \mathcal{N}(0, 0.01)^{dim(\theta_d)}$
Initialize episode turn counter $n_e \leftarrow 0$
**for** *$t$ in 1:$T$* **do**
  With probability $\epsilon$ 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 $\mathcal{D}_s$
    $n_e \leftarrow 0$
  **else**
    $n_e \leftarrow n_e + 1$
  **end**
  Sample random minibatch of transitions ($s_\tau$ , $a_\tau$, $r_\tau$, $s_{\tau+1}$) from $\mathcal{D}$
  Set $\lambda_t \leftarrow \min(\lambda, \frac{\lambda \cdot t}{k_\lambda})$
  Set $y_\tau \leftarrow \left\{ \begin{array}{ll} r_\tau - \lambda_t, & \text{for terminal } s_{\tau+1} \\ r_\tau + \max_{a'} Q(s_{\tau+1} - \lambda_t \cdot d(s_{\tau+1}; \theta_d), a_\tau, & \text{for non-terminal } s_{\tau+1} \end{array} \right\}$
  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 $\theta_d \leftarrow \theta_d - \eta \cdot \nabla_{\theta_d} \text{log\_loss}(y_j, d(s_j; \theta_d))$
**end**




(a) Adventure Seeker

(b) Cart-Pole

Figure 3: Total catastrophes for DQNs and *Intrinsic Fear* DQNs on Adventure Seeker and Cart-Pole

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

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 intrinsic fear is only applied in states from which catastrophes are inevitable, intrinsic fear doesn't help. However, if the *fear radius* is too large early on, then 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* using a linear gain until it has reached full strength at some predetermined time step $k_\lambda$. In our Cart-Pole experiments we phase in over $1M$ experiences.

## 5 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.

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 $5$. Because this problem is so simple, we phase in the fear factor quickly, reaching full strength in just $1000$ moves. On this problem we set the fear factor $\lambda$ 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.

## 6 Related Work

The paper addresses safe exploration, intrinsically motivated reinforcement learning, and the stability of Q-learning with function approximation under distributional shift.

### 6.1 The Stability of Reinforcement Learning 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 a learned strategy to avoid a specific class of catastrophic mistakes. Murata & Ozawa (2005) specifically addresses the problem of distributional shift in RL, considering the phenomena of catastrophic interference owing to function approximation with neural networks. They propose a memory-based solution. Outside of reinforcement learning, a number of papers address issues related non-stationary data distributions. (Sugiyama & Kawanabe, 2012) present a book dedicated to covariate shift. Distributional shift has been covered extensively in the context of supervised learning machine learning such as transfer learning, transduction, local learning, active learning, and semi-supervised learning (Quionero-Candela et al., 2009). Notably, Barreno et al. (2006) addresses non-stationary data in the context of security, and Szegedy et al. (2013) considers adversarial perturbations of data to fool classifiers.

### 6.2 Safe Exploration in Reinforcement Learning

Several papers consider the safe exploration for reinforcement learners. Hans et al. (2008) defines a fatal state as one from which one might receive a return below some pre-defined safety threshold. They then 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. For the backup model, the authors consider an altered Bellman equation to maximize the minimal reward to come (rather than the discounted cumulative reward). However, they do not consider the case of 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 states from which recovery is not possible, suggesting that a safe state is one from which some policy could return to the starting state $s_0$ via some return policy $\pi_r$. Shalev-Shwartz et al. (2016) considers how strong a penalty should be to discourage accidents but not the case where distributional shift dooms an agent to perpetually revisit catastrophic failure modes. They also consider hard constraints to ensure safety, an approach that differs from ours. To our knowledge, our approach differs markedly from prior work. Unlike past work, we identify and tackle potential safety issues stemming from distributional shift while using function approximation.

### 6.3 Intrinsically Motivated Agents

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); Barto et al.. The idea is that such motivation can lead agents to explore intelligently even when extrinsic rewards are sparse. Such approaches sometimes refer to this intrinsic reward for discovery as *curiosity*. Like classic work on intrinsic motivation, our methods operate by perturbing the reward function. But while these approaches assign bonuses to encourage discovery of novel transitions, we assign penalties to discourage rediscovery of catastrophic transitions.

## 7 Discussion

Our experiments suggest that absent substantial augmentation, DQNs are unacceptably brittle for use in real world applications where any harm can come of actions. While at present we show results only on toy models, likely 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 steeper angle between blade and skin. 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.

The success of our intrinsic fear models suggests 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 how precisely dangerous or catastrophic states might appear in the state space. We also assume that it is possible to encounter and detect dangerous states, without necessarily greeting catastrophe. This property needn't hold on arbitrary reinforcement learning problems. For example, in a problem where a single action, from any state, can produce a catastrophe, our methods might fail. But for real world environments with some geometric structure, this assumption seems reasonable. A self-driving car can't run over a passenger in the next second if no people are in close proximity.

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.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25. ACM, 2006.

Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016.

Justin Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. 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 et al. Policy networks with two-stage training for dialogue systems. *arXiv:1606.03152*, 2016.

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

Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. Safe exploration for reinforcement learning. In *ESANN*, pp. 143–148, 2008.

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

Sergey et al. Levine. 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. *arXiv:1608.05081*, 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*.

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. *arXiv preprint arXiv:1205.4810*, 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.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

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 preprint 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*, 3 (1):9–44, 1988.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

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