github.com/deepmind/bsuite

Behaviour Suite for Reinforcement Learning

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides,
Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari,
Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, Hado Van Hasselt

Reinforcement learning that matters

“Understand agent behaviour, develop better algorithms.”

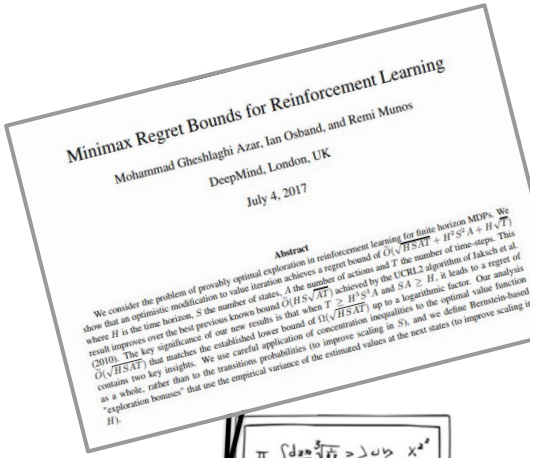
performance

safety

exploration

understanding

A spectrum of agent development



Minimax Regret Bounds for Reinforcement Learning

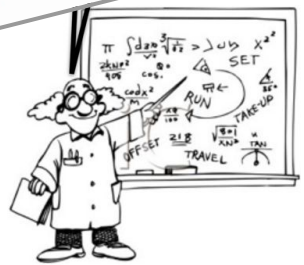
Mohammad Gheshlaghi Azar, Ian Osband, and Remi Munos

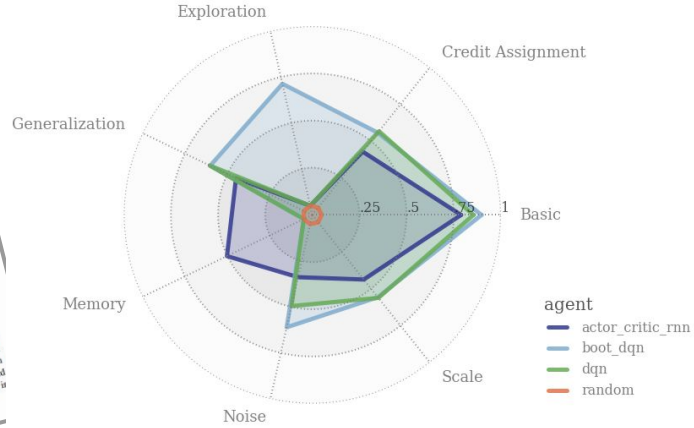
DeepMind, London, UK

July 4, 2017

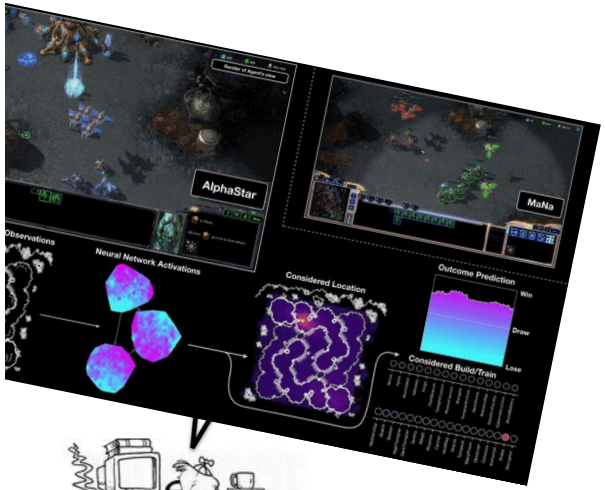
Abstract

We consider the problem of provably optimal exploration in reinforcement learning for finite horizon MDPs. We show that an optimistic modification to value iteration achieves a regret bound of $\tilde{O}(\sqrt{HSAT} + H^2S^2A + H\sqrt{T})$ where H is the time horizon, S the number of states, A the number of actions and T the number of time-steps. This result improves over the best previous known bound $\tilde{O}(HS\sqrt{AT})$ achieved by the UCRL2 algorithm of Jaksch et al. (2010). The key significance of our new results is that when $T \geq H^3S^3A$ and $SA \geq H$, it leads to a regret of $\tilde{O}(\sqrt{HSAT})$ that matches the established lower bound of $\Omega(\sqrt{HSAT})$ up to a logarithmic factor. Our analysis contains two key insights. We use careful application of concentration inequalities to the optimal value function as a whole, rather than to the transitions probabilities (to improve scaling in S), and we define Bernstein-based "exploration bonuses" that use the empirical variance of the estimated values at the next states (to improve scaling in H).





bsuite

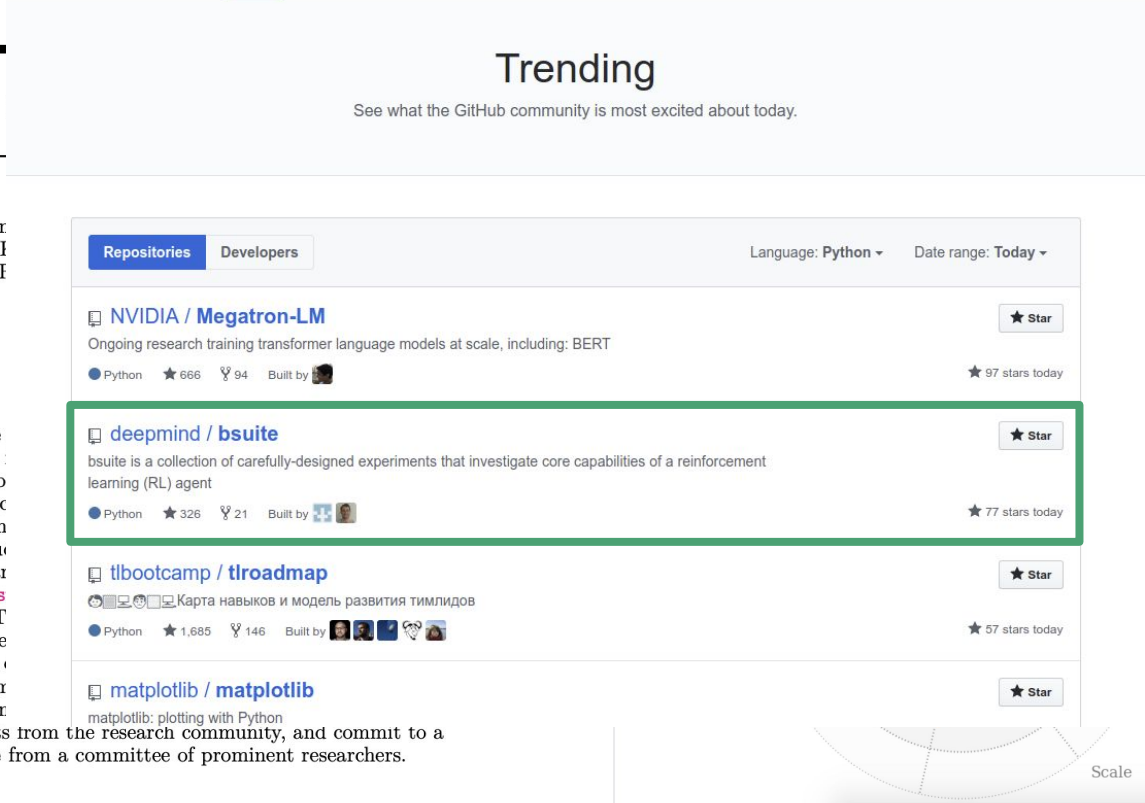


bsuite = Behaviour Suite for Reinforcement Learning

Behaviour Suite

Ian Osband, Yotar
Eren Sezener, Andre Saraiva, I
Satinder Singh, Benjamin Van I

This paper introduces the
`bsuite` for short. `bsuite`
that investigate core capab
two objectives. First, to c
that capture key issues in
gorithms. Second, to stu
on these shared benchmar
github.com/deepmind/bs
of any agent on `bsuite`. T
research on the core issue
learning algorithms. Our
projects. We include exam
as new reference implemer
more excellent experiments from the research community, and commit to a
periodic review of `bsuite` from a committee of prominent researchers.

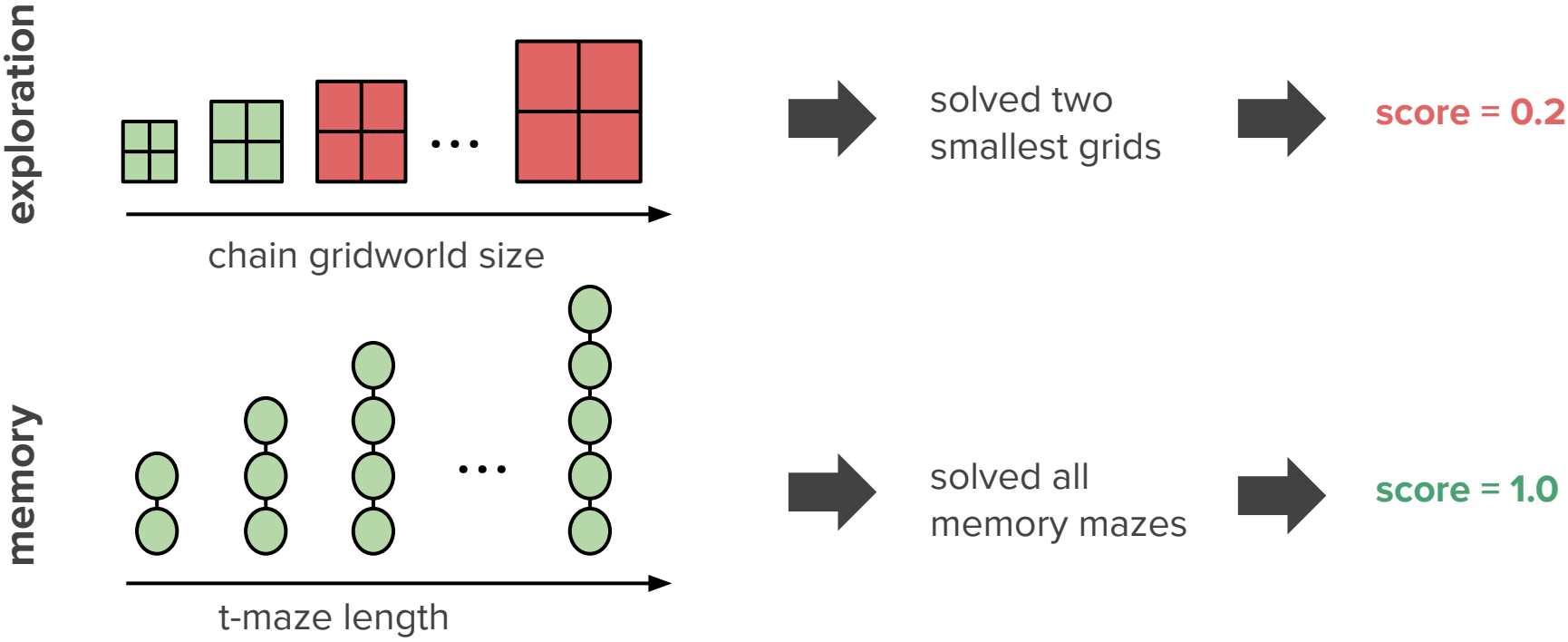


Talk outline

1. **What is bsuite?**
2. **What type of experiments go into bsuite?**
3. **Why are these experiments useful?**
4. **How can you use bsuite going forward?**

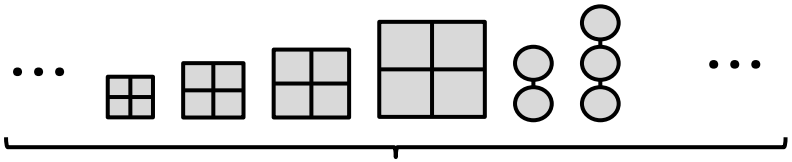
What is bsuite?

bsuite is a set of *targeted* experiments



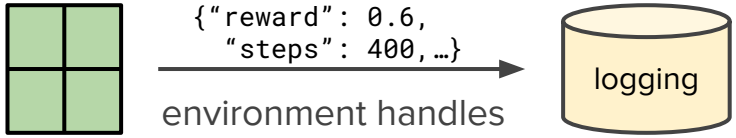
Batteries-included sweep + analysis

```
from bsuite import sweep
for bsuite_id in sweep.SWEEP:
  run(bsuite_id=bsuite_id)
```



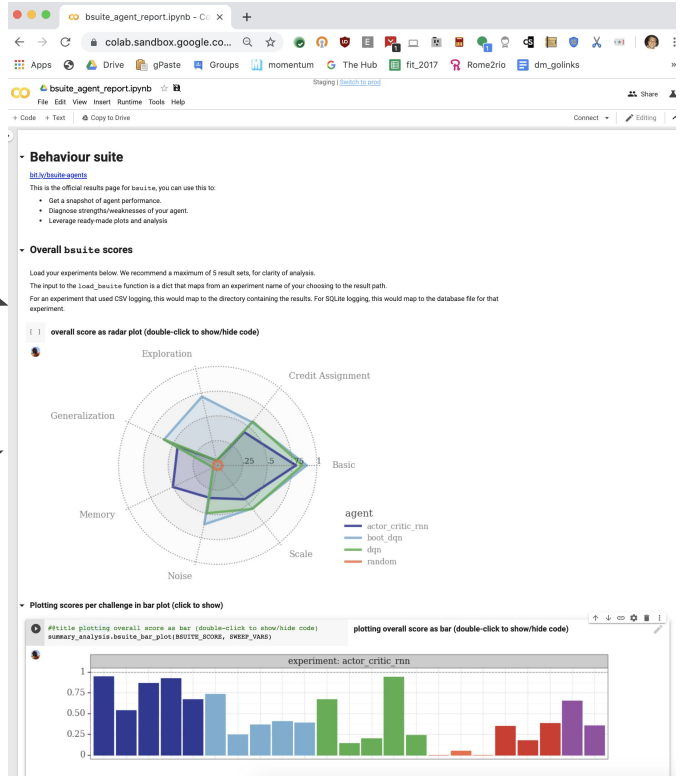
Flag **bsuite_id** specifies experiment.
Easy to combine with your code.

```
import bsuite
env = bsuite.load_and_record(FLAGS.bsuite_id)
```



environment handles
all data logging.

Automatic analysis
bit.ly/bsuite-agents

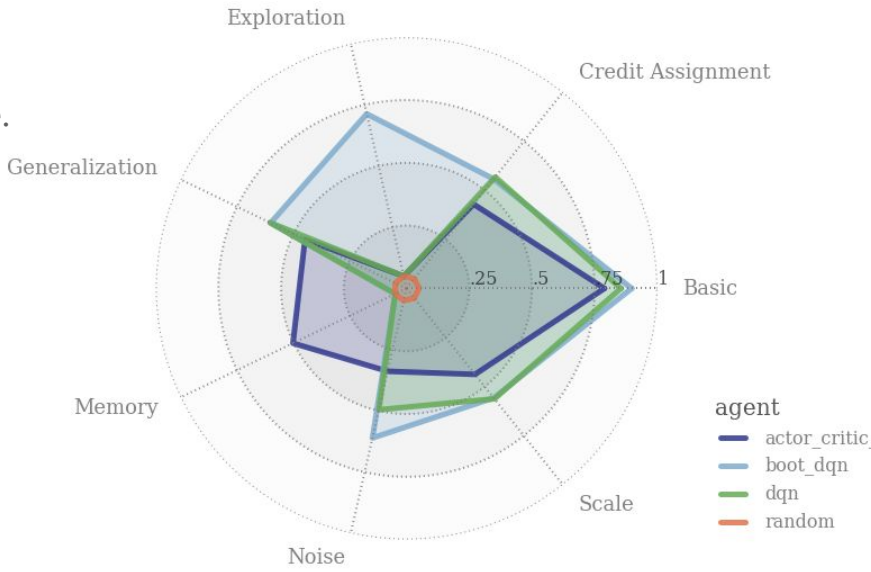


bsuite: diagnosing agent behaviour

- **Quality diagnostic experiments.**
  - **simple**: strip away confounding factors.
  - **challenging**: push agents beyond normal range.
  - **scalable**: alg scalability, not performance.
  - **fast**: iterate from launch to results in < 30min.

- **Easy to use**
  - run from **any agent framework**.
  - integrate with < 10 lines of code.
  - ready-made analysis colab.

- github.com/deepmind/bsuite



It’s **easy to use** bsuite

1. Swap out the environment

```
env = environments.load(name, settings)
```

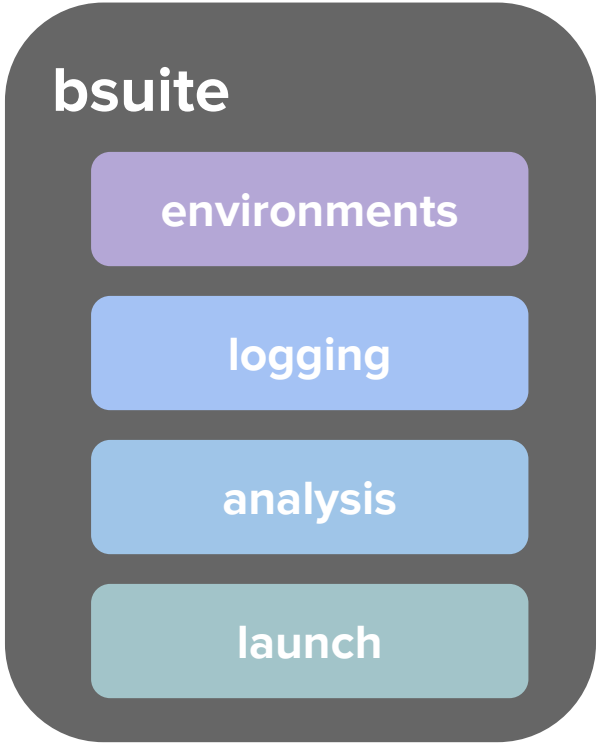
```
import bsuite

env = bsuite.load_and_record(FLAGS.bsuite_id)
```

2. Add one flag to your launch

```
from bsuite import sweep

for bsuite_id in sweep.SWEEP:
  run(bsuite_id=bsuite_id)
```



What are these experiments?

What is in a bsuite experiment?

1. **Environments**: fixed set of environments that the agent interacts with.

2. **Interaction**: fixed regime of agent/environment interaction.

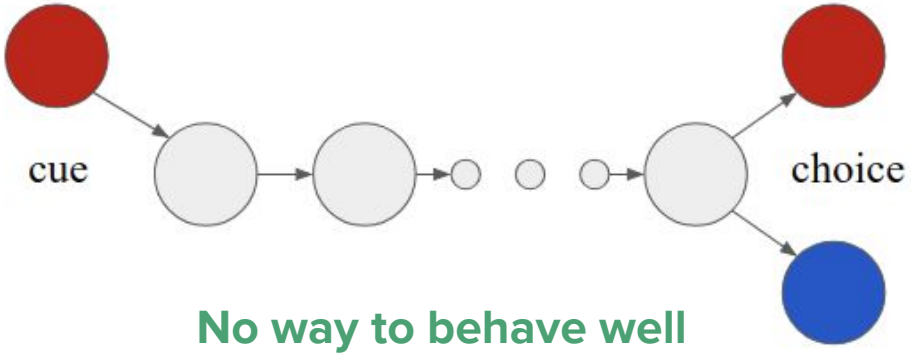
3. **Analysis**: fixed procedure mapping agent behaviour to results and plots.

Example 1: Memory length

- *“How long can the agent remember?”*

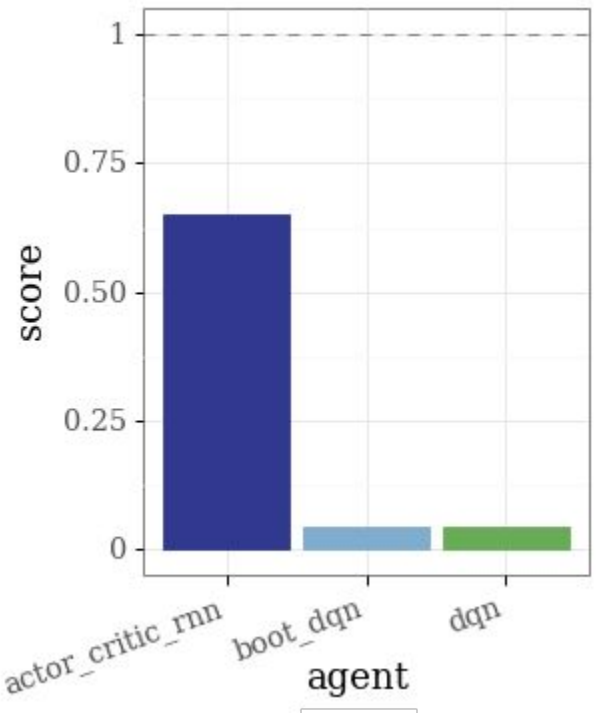
- Simple T-maze.
  - Observe context.
  - Wait N-steps.
  - Action matches context r=+1, -1

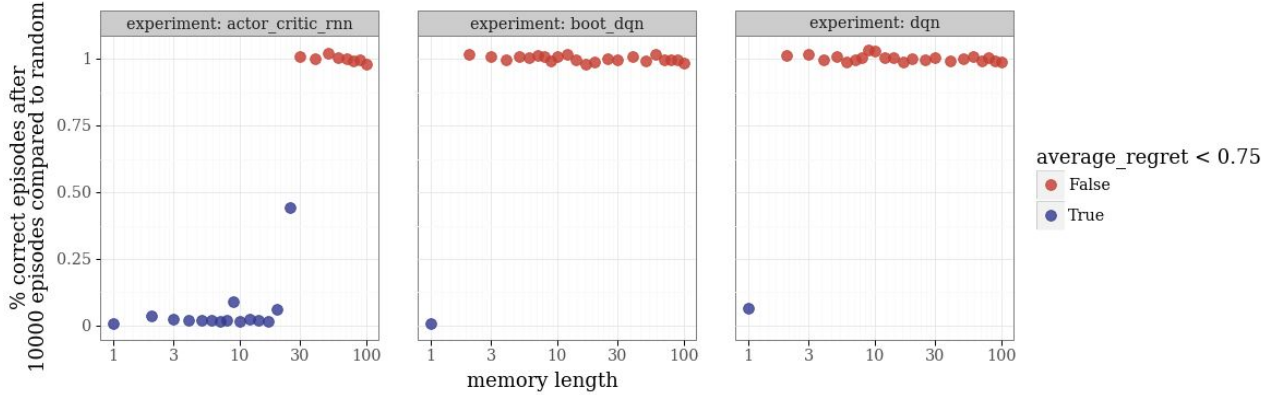
- Experiment definition
  - Run for sizes N=1..100
  - 10k episodes
  - Score = % of N where regret < 75 random



No way to behave well without memory.

Example 1: Memory length



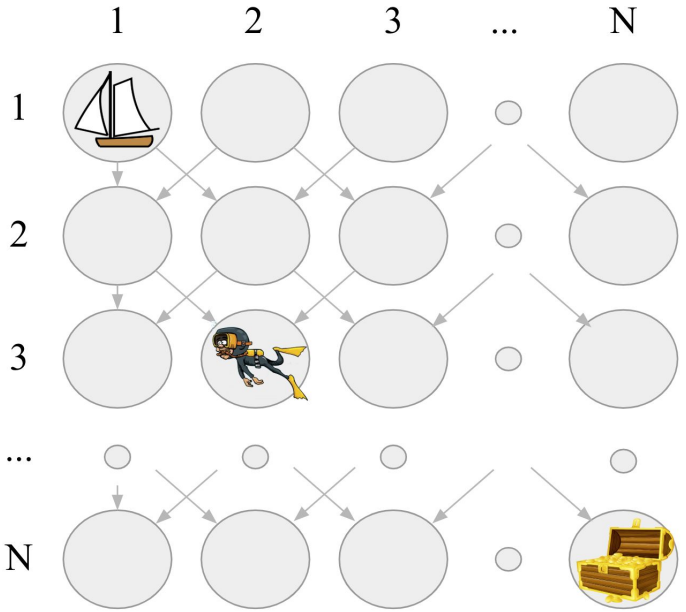


Example 2: Deep Sea

- *“Does the agent perform deep exploration?”*

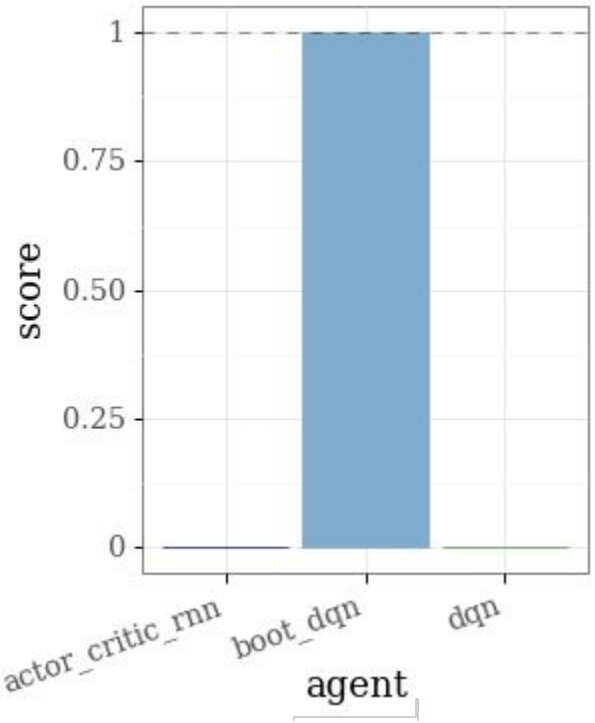
- Environment is an N x N grid.
  - Start top left, fall one row per timestep.
  - Action 0 or 1 maps to “left”/“right” by state.
  - **Small cost** for going “right”.
  - **Large reward** if you make it to the treasure.
  - **Inefficient exploration → 2^N episodes to treasure!**

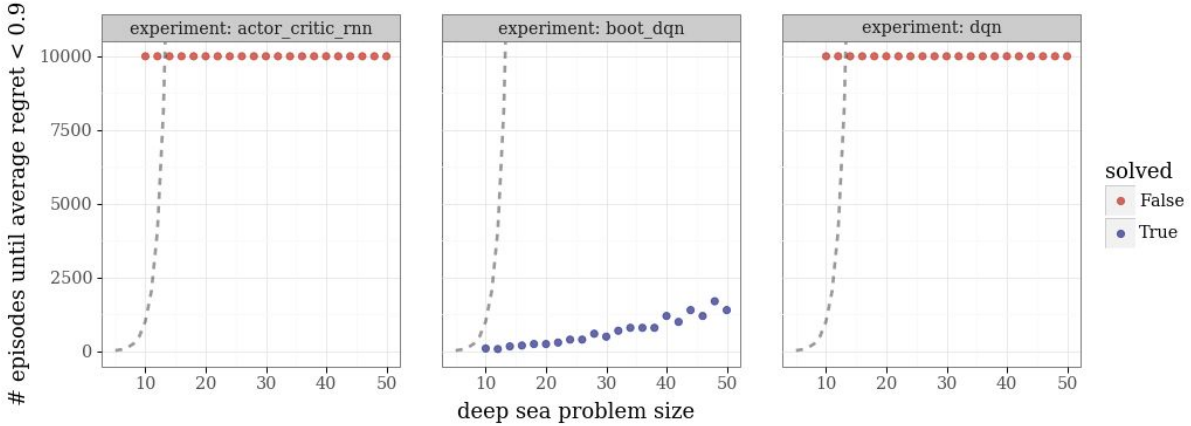
- Experiment definition:
  - Run size N=10..50 for 10k episodes
  - Score = % of N that have average regret < 0.9 (faster than 2^N baseline)



No way to behave well without exploration.

Example 2: Deep Sea

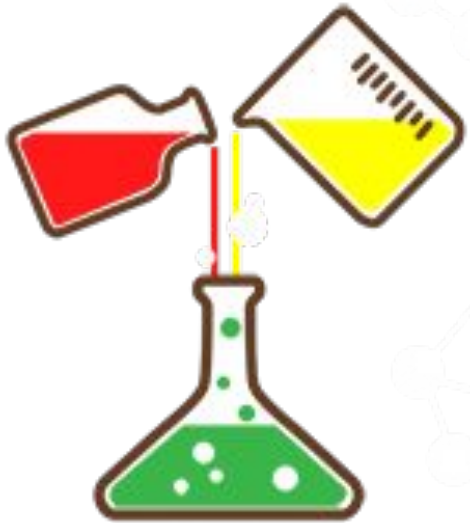




Why are these experiments useful?

Why are these experiments useful?

- Practical theory often lags **practical algorithms**.
- An “MNIST” for Reinforcement Learning.
- Open source code, **reproducible research**.
- Instantiate **key issues** in RL research.
- Valuable **“unit tests”** for core RL concepts.



How do you load the results?

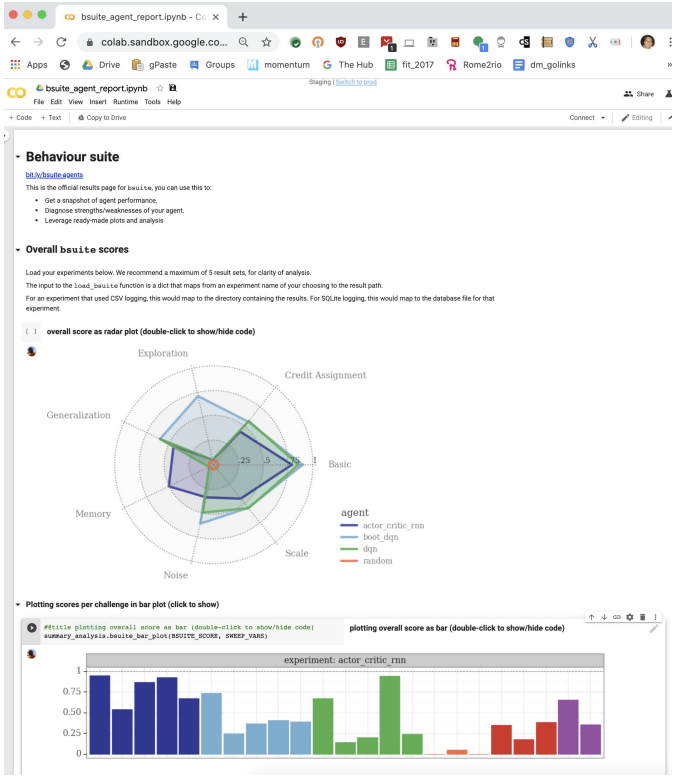
- Pre-cooked: bit.ly/bsuite-colab

```
from l.d.r.bsuite import load

XIDs = {
    'name_1': xid_1,
    'name_2': xid_2,
}
DF, SWEEP_VARS = load.load_bsuite(XIDs)
```

- DF = dataframe including results.
- SWEEP_VARS = list of hyperparams.

- **Automatically handles multiple agents and hyperparameters**.



Interactive session

- Tutorial in colab: bit.ly/bsuite-tutorial-colab
- Comparing different agents: bit.ly/bsuite-agents
- Changing optimizer in DQN: bit.ly/bsuite-optimizer
- Varying ensemble size bootstrapped DQN: bit.ly/bsuite-ensemble

How can you use bsuite?

Automated LaTeX appendix

- Run on bsuite ➔ 1-page summary.
- Easy to include in [conference format]:
  - ICML
  - ICLR
  - NeurIPS
- “Sanity check” your algorithm.
- Benchmark performance.
- Sensitivity analysis.

C `bsuite` report: benchmarking baseline agents

The *Behaviour Suite for Reinforcement Learning*, or `bsuite` for short, is a collection of carefully-designed experiments that investigate core capabilities of a reinforcement learning (RL) agent. The aim of the `bsuite` project is to collect clear, informative and scalable problems that capture key issues in the design of efficient and general learning algorithms and study agent behaviour through their performance on these shared benchmarks. This report provides a snapshot of agent performance on `bsuite2019`, obtained by running the experiments from `github.com/deepmind/bsuite` [47].

C.1 Agent definition

In this experiment all implementations are taken from `bsuite/baselines` with default configurations. We provide a brief summary of the agents run on `bsuite2019`:

- **random**: selects action uniformly at random each timestep.
- **dqn**: Deep Q-networks [38].
- **boot_dqn**: bootstrapped DQN with prior networks [46, 45].
- **actor_critic_rnn**: an actor critic with recurrent neural network [37].

C.2 Summary scores

Each `bsuite` experiment outputs a summary score in [0,1]. We aggregate these scores by according to key experiment type, according to the standard analysis notebook. A detailed analysis of each of these experiments may be found in a notebook hosted on Colaboratory `bit.ly/bsuite-agents`.

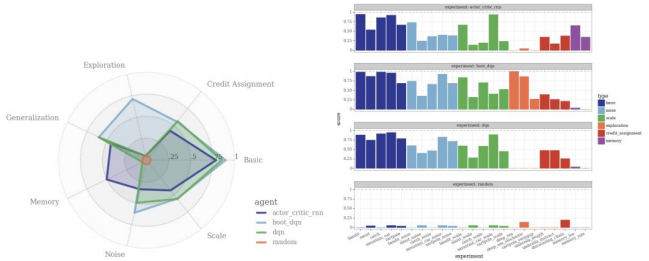


Figure 6: Radar plot gives a snapshot of agent behaviour. Figure 7: Summary score for each `bsuite` experiment.

C.3 Results commentary

- **random** performs poorly across all aspects. This confirms that our scoring functions are working as intended.
- **dqn** performs well on basic tasks, and quite well on credit assignment, generalization, noise and scale. DQN performs extremely poorly across memory and exploration tasks. The feedforward MLP has no mechanism for memory, and ϵ=5%-greedy action selection is notoriously inefficient in domains that require efficient exploration.
- **boot_dqn** performs mostly identically to DQN, except for exploration where it greatly outperforms, and also a smaller boost to performance under noise. This result matches our understanding of Bootstrapped DQN as a variant of DQN designed to estimate uncertainty and use this to guide deep exploration.
- **actor_critic_rnn** typically performs worse than either DQN or Bootstrapped DQN on all tasks apart from memory. This agent is the only one able to perform better than random due to its recurrent network architecture.

We want better experiments

- This is the start of the effort, not the end.
- **Announcing bsuite committee.**
  - Joelle Pineau (Facebook)
  - John Schulmann (OpenAI)
  - ... looking for more!
- Adding a new experiment to bsuite is easy!
- Let's talk: **@IanOsband**