The Primacy Bias in Deep Reinforcement Learning

Evgenii Nikishin [*1] Max Schwarzer [*1] Pierluca D'Oro [*1] Pierre-Luc Bacon [1] Aaron Courville [1]

- *Could overfitting on a single batch of early data be enough to entirely disrupt an agent's learning process?*

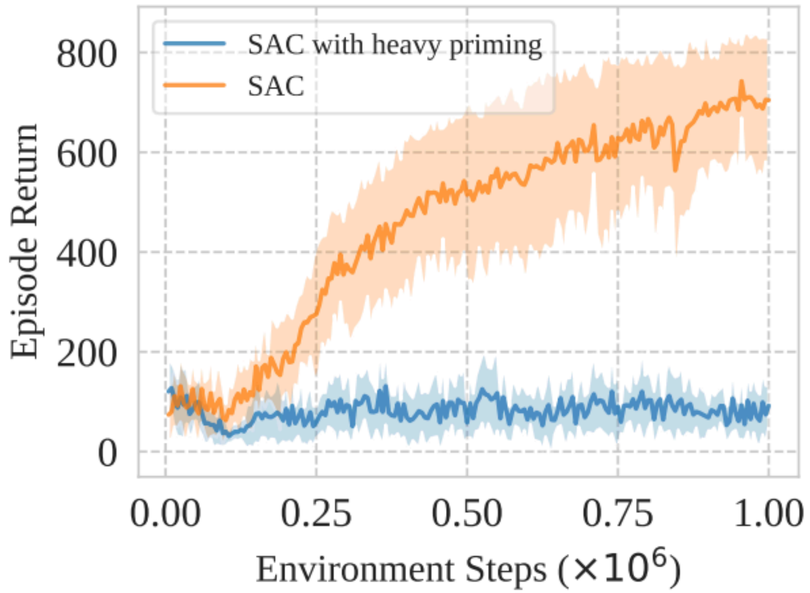


Figure 1. Undiscounted returns on `quadruped-run` for SAC with and without *heavy priming* on the first 100 transitions. An agent extremely affected by the primacy bias is unable to learn even after collecting hundreds of thousands of new transitions. Mean and std are estimated over 10 runs.

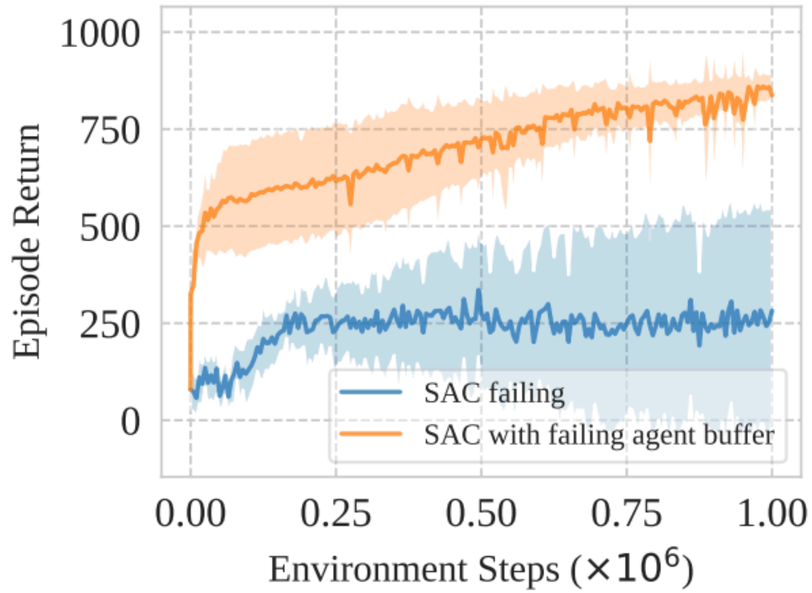


Figure 2. Undiscounted returns on `quadruped-run` for SAC trained with 9 updates per step. `SAC failing` is a standard agent; `SAC with failing agent buffer` is an agent initialized with the replay buffer of the first agent, which allows it to learn quickly. Mean and std are estimated over 10 runs.

heavy priming: after collecting 100 data points, we update the agent 10^5 times using the resulting replay buffer, before resuming standard training

Admin

- We are done with notebooks, discussion posts, tests etc
- Time to focus on your projects
- Next week we will start project standups

It’s easy to do science with computers! It’s also easy to do something that only attempts to look like science

Science, roadmaps, issues, and instances

- Today is all about methodology for empirical RL research
- Methodology: a system of methods used in a particular area of study or activity."a methodology for investigating the concept of focal points”
- If you have a clear and principled approach, you are likely to do good and valid research
  - Contrasting example: “I have a vague idea for a new algorithm, prove it is better in AIGym”
- IF you have to have good intentions, a good environment, and a good plan for your research, THEN it is likely to give the desired outcomes: knowledge & understanding

There are many kinds of RL research

- Theory
- Application, system building & challenge problems: often results in insights
- Algorithm development
- Problem formulations
- Cognitive theories
- Empirical study

▶Math

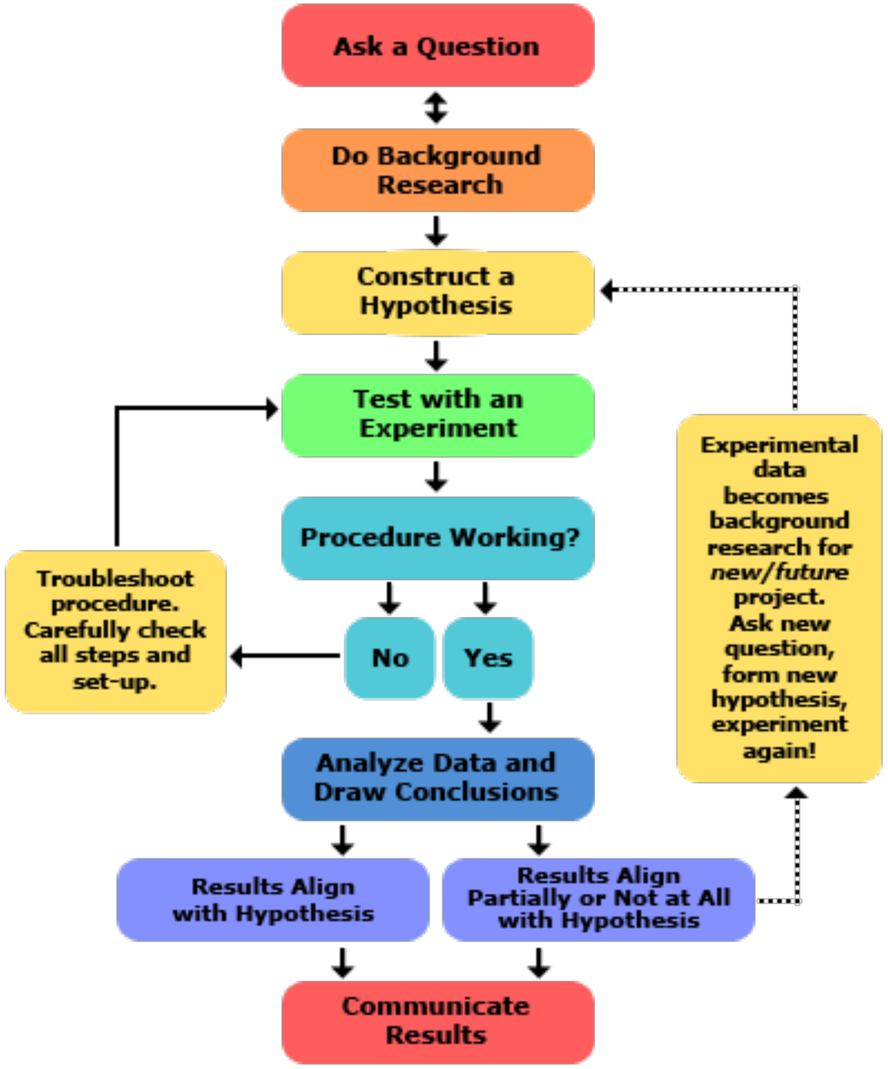
▶Engineering

▶Science

The two faces of Empirical Research in AI

- Empirical research here broadly means any activity where the primary contribution or utility is measured by experimental / numerical results
  - By running experiments
- **Dr Rainbow**: the creative, excited passionate inventor
- **Dr White**: the careful, strict, never-satisfied scientist
- You have to be both
- There is no advice on how to be **Dr Rainbow**—what ever leads to progress is OK
- Dr White does not trust **Dr Rainbow**, nor himself!!!
  - But this does not mean **Dr White's** work cannot be creative and innovative

We are in Computing Science doing RL research; we should know about the scientific method



Guidelines for empirical AI research

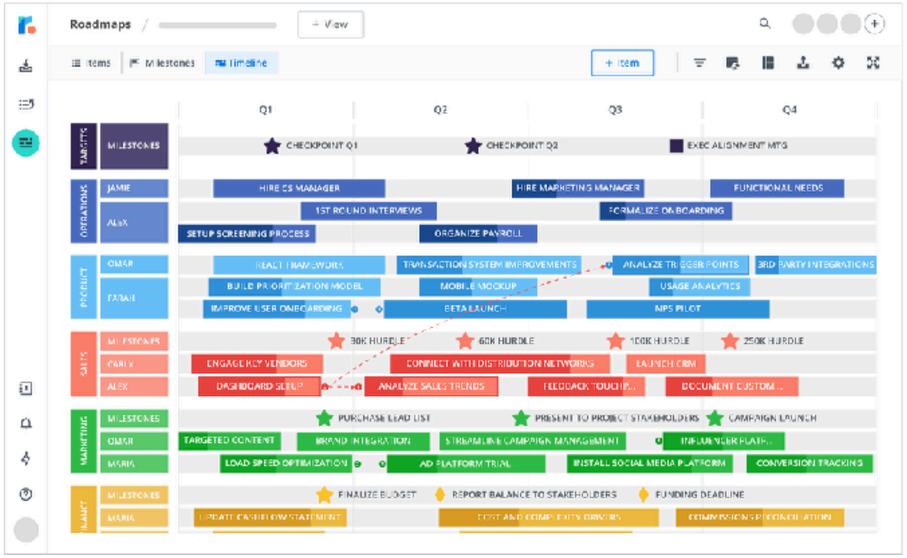
1. Embrace a roadmap
2. Define a set of solution constraints
3. Identify open questions
4. Make progress on a question
5. Communicate at all stages

- This whole thing is an iterative process of refinement, and sometimes scrapping everything and starting over

1. Embrace a roadmap

- Defines your medium and long term goals: e.g.,
  - A sketch of an AI Architecture
  - Application or challenge problem that is out of reach for current methods
  - Could just be a list of open problems
- A Roadmap changes all the time—it should
- It can be personal
- But its best used for communication and collaboration

Example roadmaps



Example roadmaps

- Planning with a learned model (topic)
- Why does distributional RL work? (question)
- Online, incremental learning with NN function approximation
- How to represent time inside the agent: recurrence and memory in state
- Developing average reward learning algorithms
- How should the agent act to balance many auxiliary losses
- How do we manage hyper-parameters

In the beginning it makes a lot of sense to follow your advisors roadmap

2. Define a set of solution constraints

- A roadmap defines a space of problems to investigate
- Solution constraints are constraints on the solution to your problems / constraints on the answers
- They sculpt how you approach each subproblem / question from your roadmap
- They depend on your ultimate goals—they are part of your roadmap
  - e.g. 1: I am interested in **linear complexity** methods, with computation i**ndependent of span**
  - e.g. 2: I want to use **all available computation and data** to reach SOTA; huge nets, lots of T-bptt, and **demonstrations** from people are all OK
- There are times when you will use solution constraints that violate those of your roadmap to make progress on a particular question

3. Identify Open Questions

- Given a roadmap and a set of solution constraints, we can pick off project sized-questions/problems
  - Good time to **iterate again** on your roadmap
  - And you are not beholden to your roadmap
- An issue is a problem that can be stated as a research question (we like questions in science). E.g.,:
  - Can we improve an existing solution according to some metric (speed)?
  - Can we come up with a new exploration method? progress on a higher-level challenge
  - What causes *Catastrophic Interference?* Study a specific phenomenon (see French's work)
  - Why did SAC do X in problem Y? Exploring unexpected results from prior work
  - How do we mitigate divergence with the Deadly Triad? Remove an undesirable property

Minimal instances (aka Diagnostic MDPs)

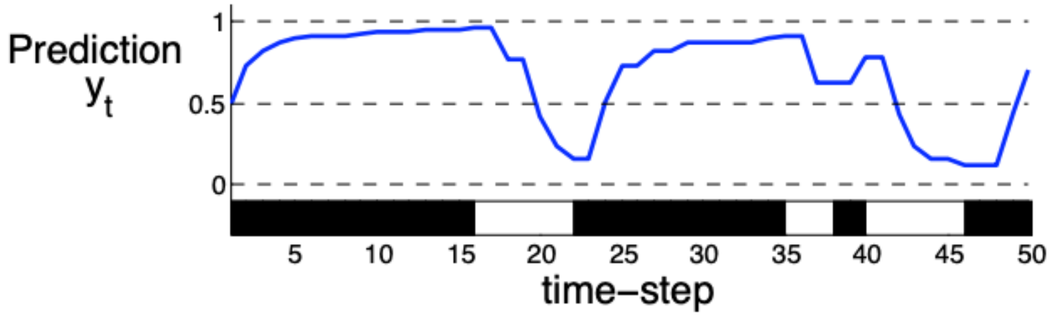
- Ideally you can make a small problem (experiment) that represents the issue you are working on
- Solving that problem demonstrates progress on your issue
- These should be small; allow extensive experimentation and analysis
  - Lots of runs, lots of hypers, very very fast
- There should be clear baselines—including ones that don't fit your solution constraints:
  - e.g., LSTD can solve the deadly triad, but requires quadratic computation

Minimal instances

- Sutton's **Black and White World**: need for tracking in partially observable tasks

On the Role of Tracking in Stationary Environments

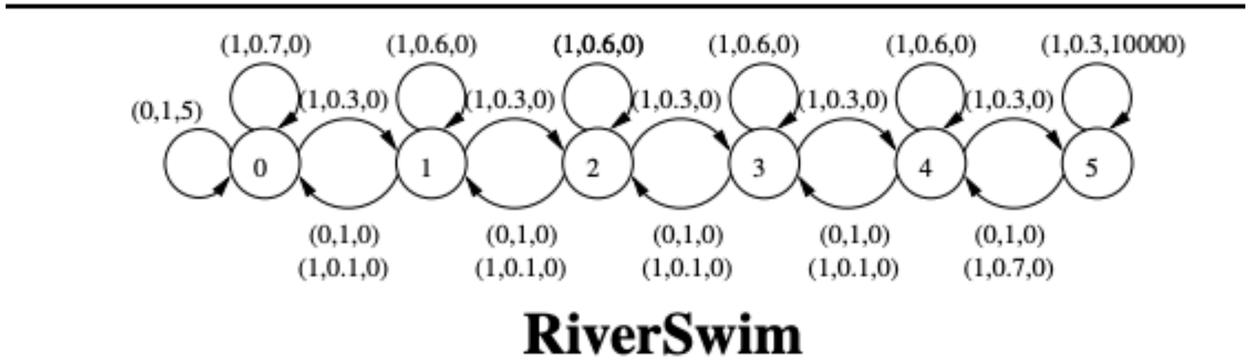
Figure 1. The Black and White world. The agent follows a random walk right and left, occasionally observing the color above it. The states wrap.



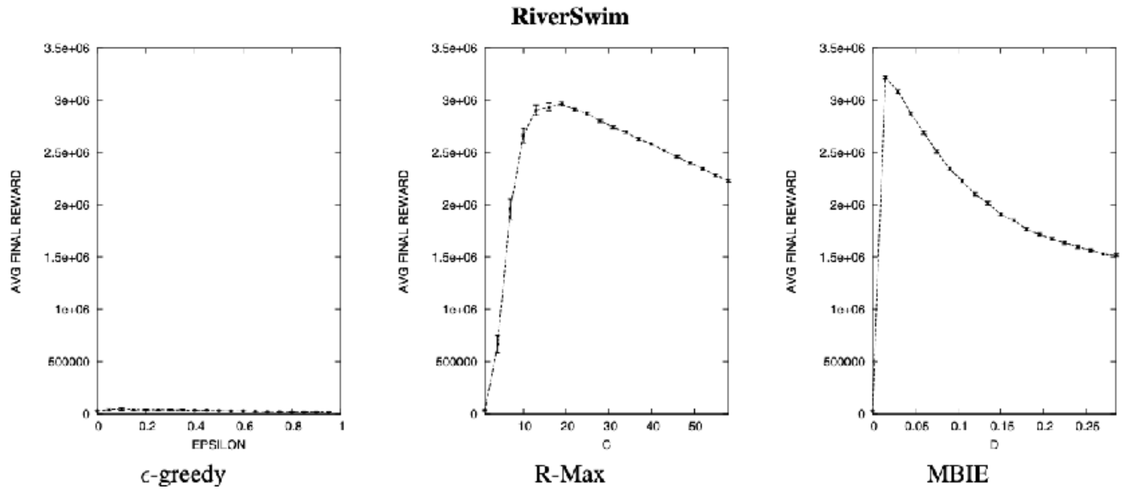
http://www.incompleteideas.net/papers/SKS-07.pdf

Minimal instances

- Strehl and Littman's work on **RiverSwim**: need for persistent exploration



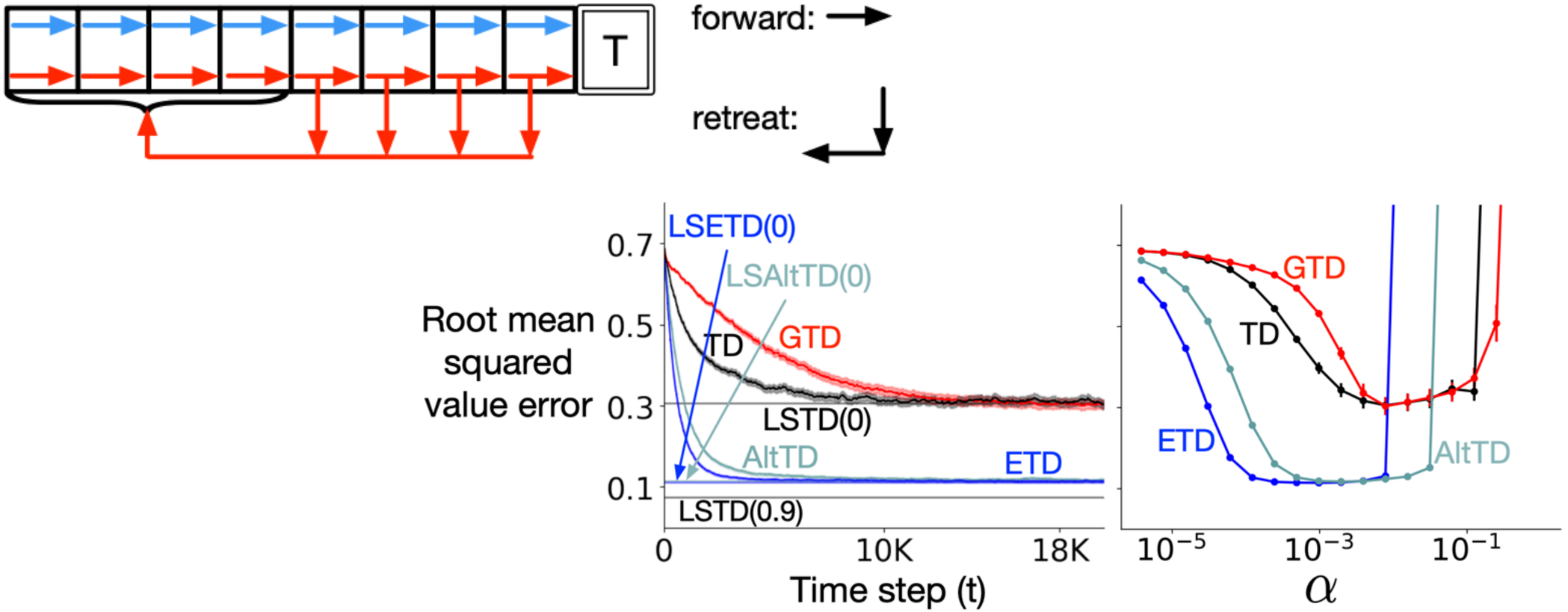
RiverSwim



https://ieeexplore.ieee.org/abstract/document/1374179?casa_token=diwCjuqafskAAAAA:X7GqWg8yM04MDfS0zAStXR5FAmkYOVIHm5d802P3uAlNMSMipVzfON9VcRMjTPVYDwRUKBe1

Minimal instances

- Ghiassian's **Collision Task**: advantages of Emphatic methods for prediction



https://arxiv.org/pdf/1811.02597.pdf

4. Make progress on a question

- There is no advice for solving problem :P
- But evaluating progress is where science and good experiments come in
- Your progress might take many different forms
- A first demonstration of something:
  - First linear complexity off-policy TD method with FA (theory & solve Baird's)
  - Improved exploration on Pitfall!
  - A new method that can adapt lambda of TD methods (theory and experimental evidence)

Progress might take many different forms …

- Performance comparable to baselines (baselines that do not respect solution constraints)
- E.g.,
  - Linear off-policy TD learning comparable to LSTD which is quadratic
  - Recurrent learning system that achieves comparable performance to RNNs with truncated backdrop through time, but with less memory & compute
  - Model-free exploration comparable to model-based approaches

Progress might take many different forms …

- Improvement over baselines that **DO respect** solution constraints
- E.g.,
  - Off-policy learning with less variance than gradient TD methods
  - Improving Performance in Reinforcement Learning by Breaking Generalization in Neural Networks.
  - Improving step-size parameter adaption: Meta-descent for online, continual prediction

Progress might take many different forms …

- New empirical insights
- E.g.,
  - Adapting behaviour via intrinsic reward: A survey and empirical study
  - Prediction in Intelligence: An Empirical Comparison of Off-policy Algorithms on Robots
- In all cases, progress should feed back into your roadmap

5. Communicate all the time

- It’s never to early to present in a lab meeting, to your research group, etc
- Talk about the problem formulation and related work:
  - **Outcome**: people help you find other refs and better formulations
- Talk about the basic approach & initial ideas for experiments:
  - **Outcome**: people have other ideas and help you avoid wasting your time
- Talk about initial results and where you are stuck:
  - **Outcome**: people suggest insightful experiments, ideas to unstick you
- Talks in general: 20mins to 1hr where several people think hard about your problem and offer free advice!

Communicate with yourself: Write in your notebook

Guidelines for empirical AI research

1. Embrace a roadmap
2. Define a set of solution constraints
3. Identify open questions
4. Make progress on a question
5. Communicate at all stages

- This whole thing is an iterative process of refinement, and sometimes scrapping everything and starting over

Don’t conflate the problems and solutions

- Always clearly separate problem and solution
- Decide on a problem setting … fix it! Don’t change it
  - Then think about solutions!
  - (How does a roadmap help with this)
- Researchers often iterate the problem of study and the solution method:
  - This results in making the problem fit your method (bad)
  - And generally coming up with solutions that don’t match your original question
  - e.g., developing new method for your concrete instance, finding it doesn’t work, and then making the concrete instance easier

Different approaches to algorithm development

- **Read and make connections**
  - See gaps, see opportunities, notice when someone solves a sub-problem differently then you would
- **The Derivation approach**
  - New loss -> new alg
  - New problem formulation, theory, new alg (common in bandit literature)
- **The mechanical approach (i.e. taking a Computational view)**
  - Under stand the issue very deeply via a minimal instance
    - Called “being the agent”
  - Incrementally test and understand the limitations of small algorithmic modifications

The mechanical approach (i.e. taking a Computational view)

A case study …

Finding answers / designing solutions

- Read a lot!
- Don’t ignore theory: as Csaba says: ask a theory guy sometimes they know the answer before you conduct the experiment
- Use implementations to understand the issue better!
- Be the agent
- Be paranoid: assume you have bugs, focus on correctness over performance at first
- Probe and test the agent: ask it questions
- Change one thing at a time!

Don’t get attached to your solution/approach/idea

Designing and running an experiment

- Write down hypothesis first
- Design the evaluation scheme and results pipeline
- Think of the configuration space:
  - Hyper-parameters, runs (seeds), baselines, problem variations etc
- Make comparisons as comparable as possible: e.g., in policy evaluation all methods can use the exact same training data—don’t leave these things to chance
- Define success

- All these should match your solution constraints

Think about the robustness of your approach

- Do the basic conclusions hold in the face of:
  - Noise: many seeds, many start states, sometimes noisy dynamics
  - Environment and training setup: multiple environments reduce chances of *overfitting*
  - Architect design and architecture choices: different optimizers, feature construction methods, exploration mechanisms, hyper-parameters
- Domain overfitting in RL is a challenge (see https://www.cs.utexas.edu/~pstone/Papers/bib2html-links/ADPRL11-shimon.pdf)
- This is why we have agent's that can play Atari well, but are difficult to adjust to gridworlds and other small domains

Thoughts on experiment workflow

- Write tests
- Run small experiments to probe correctness and if the idea has hope
- Use small benchmarks as a sanity check (e.g., mountain car)
- Use python notebooks as a lab book
- Log negative results
- Document everything in your Lab Book or your research journal
- Save everything: don’t get into the situation where plotting something different requires re-running everything
- Don’t over-generalize the software: KISS, research code is throw-away code

If your experiment produces a result you don't expect …

STOP! DONT CHANGE ANYTHING

Figure out why your program produced that result first

Projects

- Your Questions for me!

    - Status up anyone?