Dr. Phil Winder

Winder
ML | RL | MLOps | DATA SCIENCE

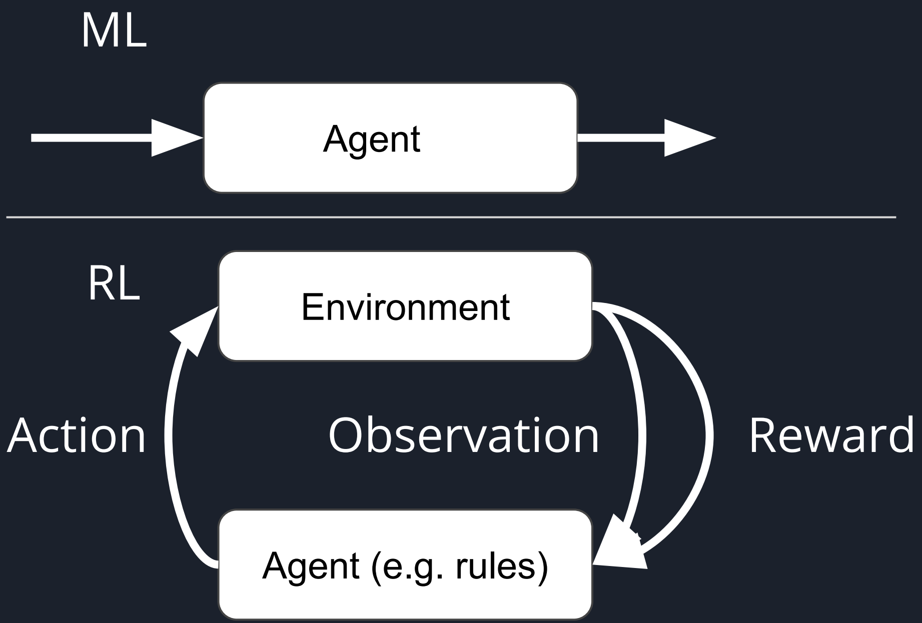
REINFORCEMENT LEARNING:

ReFrame: A Process for Finding and Executing RL Projects

in LIVE

Reinforcement Learning Recap

For one Episode



https://covariant.ai/

https://winder.ai

What is an RL Problem?

- Sequential – iterative
- Strategic – course-corrections
- Long-term rewards
- Actions affect the future
- Actions affect the reward

What Are You Looking For?

- An entity performing an action: a person, a PID controller, an advanced process control system
- An environment: a well-bounded context
- A clean interface between the entity and the environment: an API, a lever, a button
- An environment that changes state: affected by the entity
- An action being performed by the entity: rules of thumb, gut feeling, experience, like riding a bike
- Some kind of success or failure: profit, KPIs, optimal temperatures, likes

But What If?

- Episode is 1 step, labelled data
    - Supervised machine learning
- Episode is 1 step, no data
    - One-state Markov chain - Multi-Arm Bandits
- Cannot affect environment & fully observable
    - Markov Chain - Use Monte Carlo techniques
- Cannot affect environment & NOT fully observable
    - Non-Markovian - Hidden Markov Models

Remember:

- Multi-step, long-term rewards, agent affects environment & outcome

The Process

1. Environment engineering
2. State observation engineering
3. Policy engineering
4. Reward engineering
5. Deployment
6. Repeat

Environment Engineering - i.e. Simulation

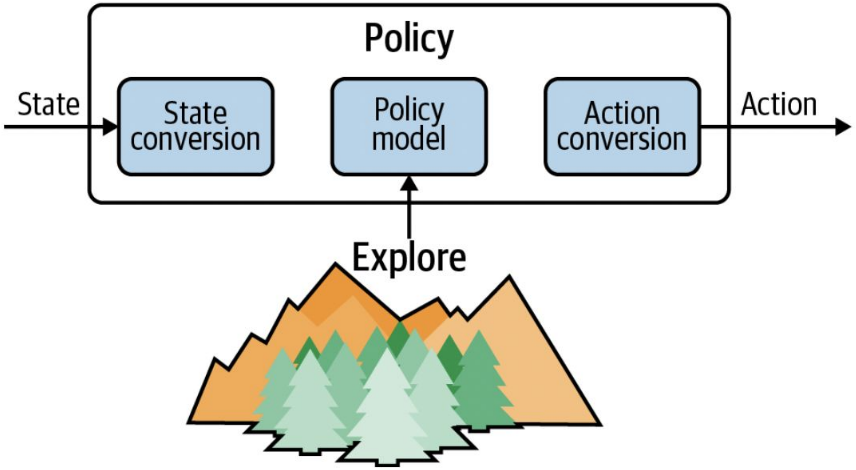
- Most profitable first step
  - EDA enhances understanding
  - Quickly find potential issues with RL solution
  - Can use random agents to explore environment, prove that it works
- Where do environments come from?
  - Physical models / Simulations
    - Augmented versions of
  - Data-driven models
    - Statistical approximations
    - Model-based Approximations
  - Generative models
  - Real-life

State/Observation Engineering

- NOT feature engineering (i.e. not creating simpler features)
- NOT policy engineering
- Better representations of the state
  - Which may include feature engineering :-)
- Domain expertise is paramount
  - E.g. robot gripper - camera vs "height of gripper"
- How?
  - Learn a forward model - lab experiments
  - Apply constraints, smaller state spaces are faster to learn
  - Dimensionality reduction (to reduce the state space)
  - Experimentation

Policy Engineering - Observations

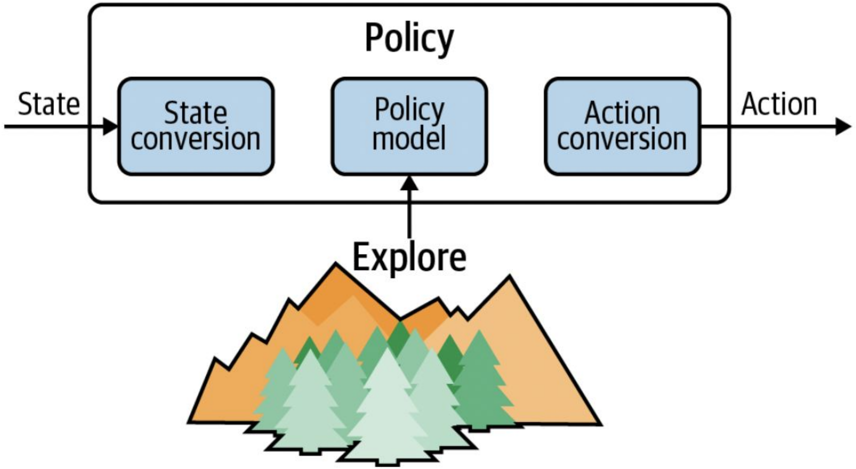
- Observations and actions need conversion
- Observations:
  - Discrete states are easier to solve - no "in-between" states
  - Continuous states harder to solve - infinite real values - no convergence guarantee
    - I.e. models must approximate - i.e. you need an ml algorithm
- Discretisation
  - Binning, tile coding, hashing, classification, unsupervised methods, etc.



s://winder.ai

Policy Engineering - Actions

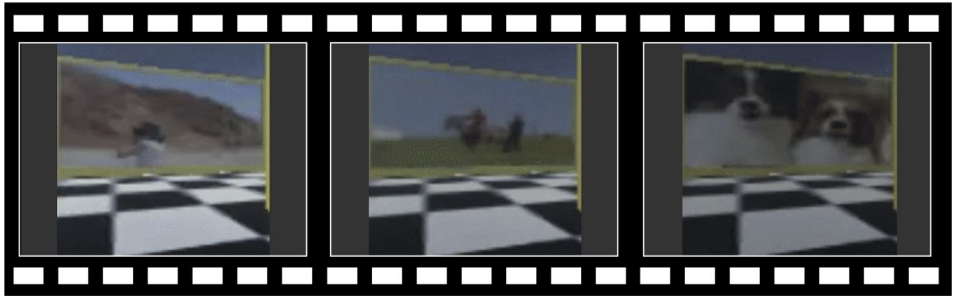
- Observations and actions need conversion
- Actions:
  - Binary - easy to work with
  - Continuous - often modelled as a random variable to aid exploration
  - No action - options framework
  - Ranked options
- Recommendations
  - Lots of diversity in Implementation
  - Many algorithms expect a certain type of data
  - Try to stick to one type
  - At least start with something simple

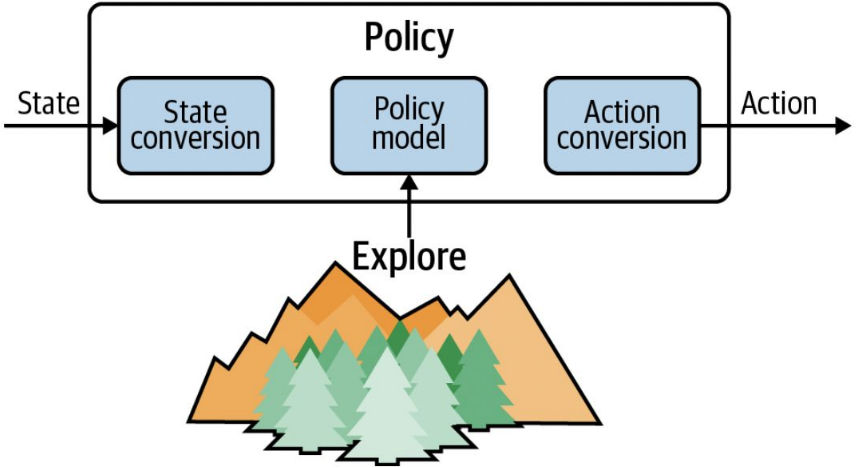


s://winder.ai

Policy Engineering - Exploration

- Lots of diversity in implementation
  - Stick with something simple to begin with
    - E.g. epsilon greedy, or whatever the algo utilises, e.g. entropy
- Kids - depth first, then transfer
  - RL - stumbling
- More advanced:
  - Info gain (surprise) - E.g. use of entropy in SAC
  - State prediction (self-reflection)
  - Random distillation (novelty)
  - Episodic curiosity (distance to novelty)
  - Curriculum learning - (teaching)



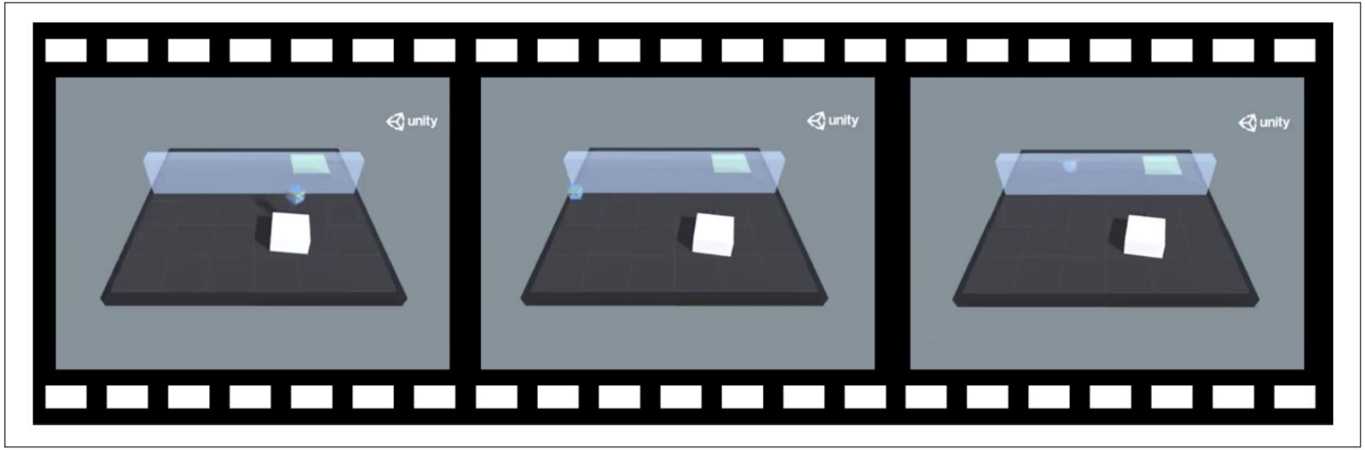


s://winder.ai

Reward Engineering

- Match the business problem
- Proxy rewards correlate with the business problem
- Are Not noisy
- Include non-functional requirements
- Are provided quickly
- Avoid plateaus
- Smooth
- Fast to compute

- Most of all - Are simple



ps://winder.ai

Reward Engineering - Common Reward Types

- Sparse rewards
- Distance to goal
- Punishing steps
- Punishing damaging or dangerous behaviour
- Goal states - e.g. targets, images

https://winder.ai

Winder.AI
ML | RL | MLOps - Consulting | Development

Summary

- Multi-step, long-term rewards, agent affects environment & outcome
- Simulations are useful
- Lots of engineering still to be done
- Rewards are hard

- Next time: Key challenges to watch out for!

https://Winder.AI/events/
phil@winder.ai
DrPhilWinder

https://winder.ai

Winder.AI
ML | RL | MLOps - Consulting | Development

Dr. Phil Winder

Winder
ML | RL | MLOps | DATA SCIENCE

REINFORCEMENT LEARNING:

ReFrame Pt. 2: How We Overcame Key RL Challenges

in LIVE -ish

Last Event Recap

- Multi-step, long-term rewards, agent affects environment & outcome
- Simulations are useful
- Lots of engineering still to be done
- Rewards are hard

Challenge 1: Framing the Problem

- Imagine yourself trying to solve it? How would you learn? What's missing?
- *Simplify* the task as much as possible, then keep iterating.
- Is there a hierarchy to the problem? Could you split it up
- Think about history, do you need to remember what you did? If yes, can you think of actions that removes the need for having history?

Challenge 2: The Environment

- Often hard to develop in “real life” - develop a simulator
- Easy to over-complicate simulators
- Develop multiple simulators
  - With varying degrees of difficulty
  - Stressing different problems within the environment

Challenge 2: Rewards

- Scale - super important, especially when you have competing concerns
- Clipping - try to avoid throwing away info
- Complex “models” or transformations of reward
- Using known models - e.g. robot must stand

Challenge 3: Training & Development

- Start simple
- Create baseline agents
  - Random
  - Fixed pick a single action (e.g. most popular)
  - Simpler RL algorithms like MCMC or Cross Entropy Method
- Create regression tests
  - Develop regression tests for situations where “it must get it right”
- Beware of long training durations
- Keep track of your experiments
- Randomness - averaging, seeds, stability, etc.
- Sensitivity to hyper parameters

Challenge 4: Evaluation

- Be careful, visualise
- Algorithmic performance improvements aren't everything
- Many sources of stochasticity
- Which leads to potentially damaging outliers

https://winder.ai

Winder.AI
ML | RL | MLOps - Consulting | Development

Challenge 5: Deployment

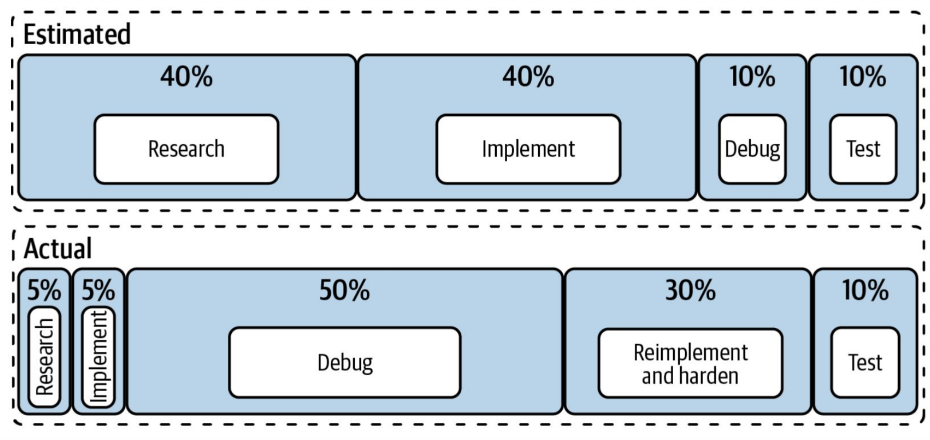
- Deployment options are immature - expect a lot of engineering

https://winder.ai

Winder.AI
ML | RL | MLOps - Consulting | Development

Challenge 6: Debugging

- Debugging is hard -
  - one study showed that code-level optimizations improved performance more than the choice of algo
  - Another showed how a single line bug (zeroing an array) caused oscillation in the value estimates
- Standard software engineering debugging techniques are useful
- Monitoring training metrics, evaluation provide the ability to experiment
- If in doubt, start with something simpler
- Most modern “state-of-the-art” algos are hardware optimisations
- Apps “fail” because the problem isn’t suited to RL



https://winder.ai

Summary

- Many challenges!
- KISS - Iterate, don’t jump
- Simulations help ease development pain, even if they’re not perfect
-

https://winder.ai

https://Winder.AI/events/
phil@winder.ai
DrPhilWinder

Winder.AI
ML | RL | MLOps - Consulting | Development