

Reinforcement Learning and its Inverse: Applications for Quantitative Finance

September 2020

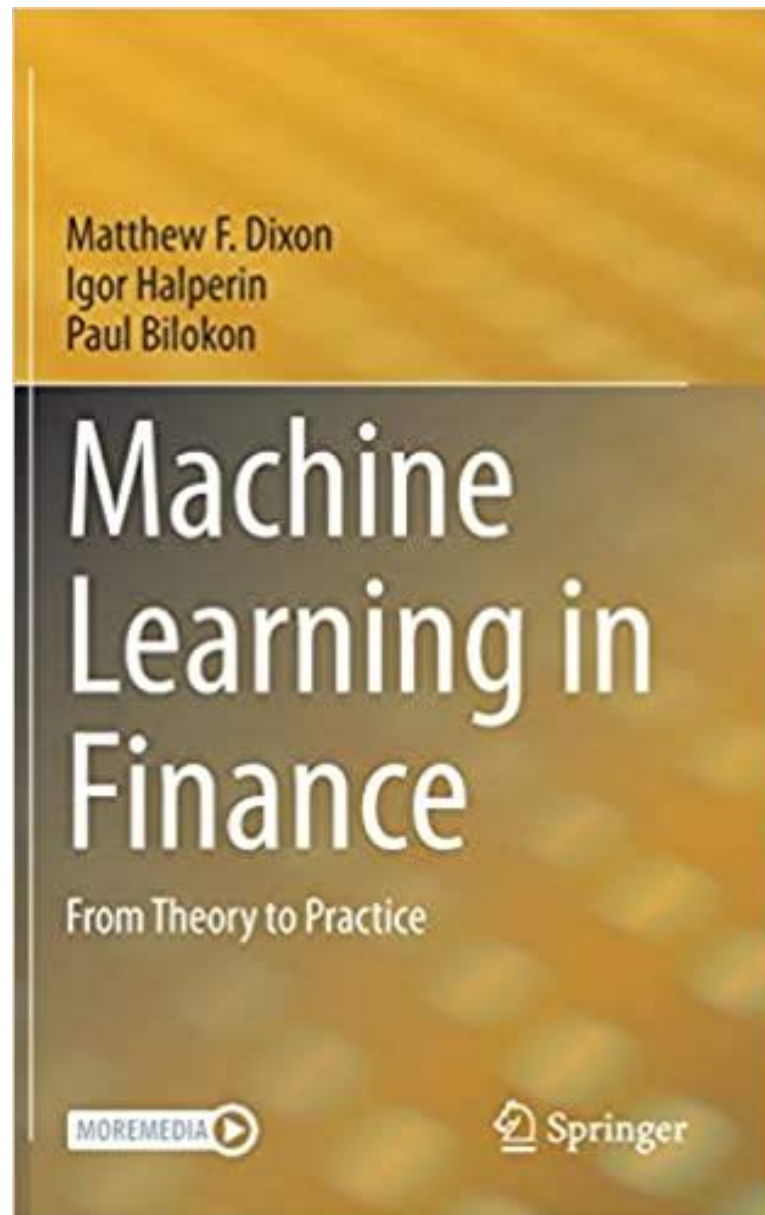
Igor Halperin

NYU Tandon School of Engineering

Disclaimer

This presentation was produced solely by Igor Halperin. The opinions and statements expressed herein are those of Igor Halperin and are not necessarily the opinions of any other entity, including UBS AG and its affiliates. UBS AG and its affiliates accept no responsibility whatsoever for the accuracy, reliability or completeness of the information, statements or opinions contained in this presentation and will not be liable either directly or indirectly for any consequences, including any loss or damage, arising out of the use of or reliance on this presentation or any part thereof. Reproduced with permission.

MLF: RL and IRL chapters



Part III Sequential Data with Decision-Making

9	Introduction to Reinforcement learning	307
1	Introduction	308
2	Elements of reinforcement learning	312
2.1	Rewards	313
2.2	Value and policy functions	314
2.3	Observable versus Partially Observable Environments	315
3	Markov Decision Processes	318
3.1	Decision policies	320
3.2	Value functions and Bellman equations	322
3.3	Optimal policy and Bellman optimality	325
4	Dynamic programming methods	328
4.1	Policy evaluation	329
4.2	Policy iteration	331
4.3	Value iteration	332
5	Reinforcement learning methods	334
5.1	Monte Carlo methods	336
5.2	Policy-based learning	338
5.3	Temporal difference learning	340
5.4	SARSA and Q-learning	342
5.5	Stochastic approximations and batch-mode Q-learning	346
5.6	Q-learning in a continuous space: function approximation	353
5.7	Batch-mode Q-learning	357
5.8	Least Squares Policy Iteration	361
5.9	Deep reinforcement learning	364
6	Summary	366
7	Exercises	368
8	References	375

11	Inverse Reinforcement Learning and Imitation Learning	453
1	Introduction	454
2	Inverse Reinforcement Learning	457
2.1	RL versus IRL	459
2.2	What are the criteria for success in IRL?	461
2.3	Can a truly portable reward function be learned with IRL?	462
3	Maximum Entropy Inverse Reinforcement Learning	463
3.1	Maximum Entropy principle	465
3.2	Maximum Causal Entropy	468
3.3	G-learning and soft Q-learning	470
3.4	Maximum Entropy IRL	473
3.5	Estimating the partition function	476
4	Example: MaxEnt IRL for inference of customer preferences	478
4.1	IRL and the problem of customer choice	478
4.2	Customer utility function	479
4.3	Maximum Entropy IRL for customer utility	481
4.4	How much data is needed?: IRL and observational noise	484
4.5	Counterfactual simulations	486
4.6	Finite-sample properties of MLE estimators	488
4.7	Discussion	491
5	Adversarial Imitation Learning and IRL	493
5.1	Imitation learning	493
5.2	GAIL: Generative Adversarial Imitation Learning	494
5.3	GAIL as an art of bypassing RL in IRL	496
5.4	Practical regularization in GAIL	499
5.5	Adversarial training in GAIL	501
5.6	Other adversarial approaches*	503
5.7	f-divergence training*	503
5.8	Wasserstein GAN*	504
5.9	Least Squares GAN*	505
6	Beyond GAIL: AIRL, f-MAX, FAIRL, RS-GAIL, etc.*	506
6.1	AIRL: Adversarial Inverse Reinforcement Learning	507
6.2	Forward KL or backward KL?	509
6.3	f-MAX	510
6.4	Forward KL: FAIRL	512
6.5	Risk-sensitive GAIL (RS-GAIL)	514
6.6	Summary	515
7	Gaussian Process Inverse Reinforcement Learning	516

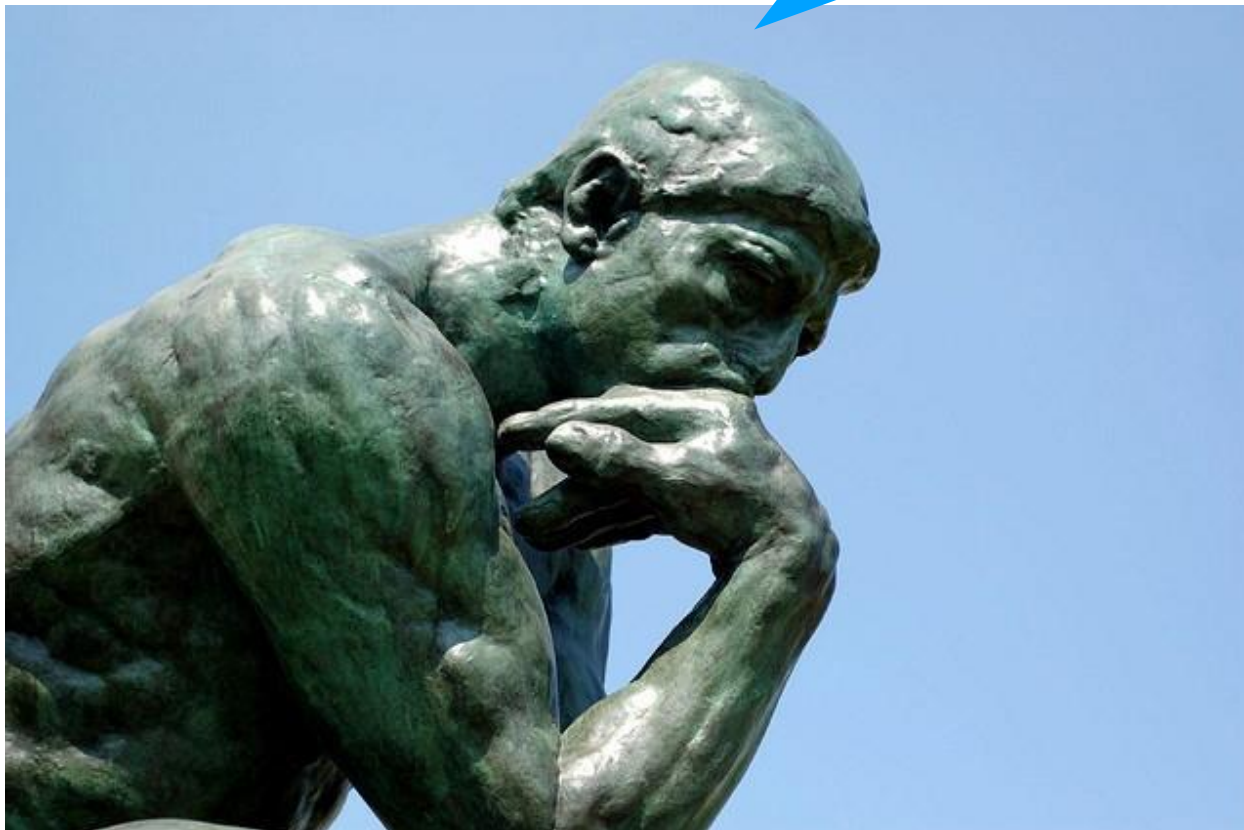
The Vapnik principles

“There is nothing more practical than a good theory.”
(Vladimir Vapnik)

“One should avoid solving more difficult intermediate problems when solving a target problem.”
(Vladimir Vapnik).

RL or IRL?

AI or not
AI?



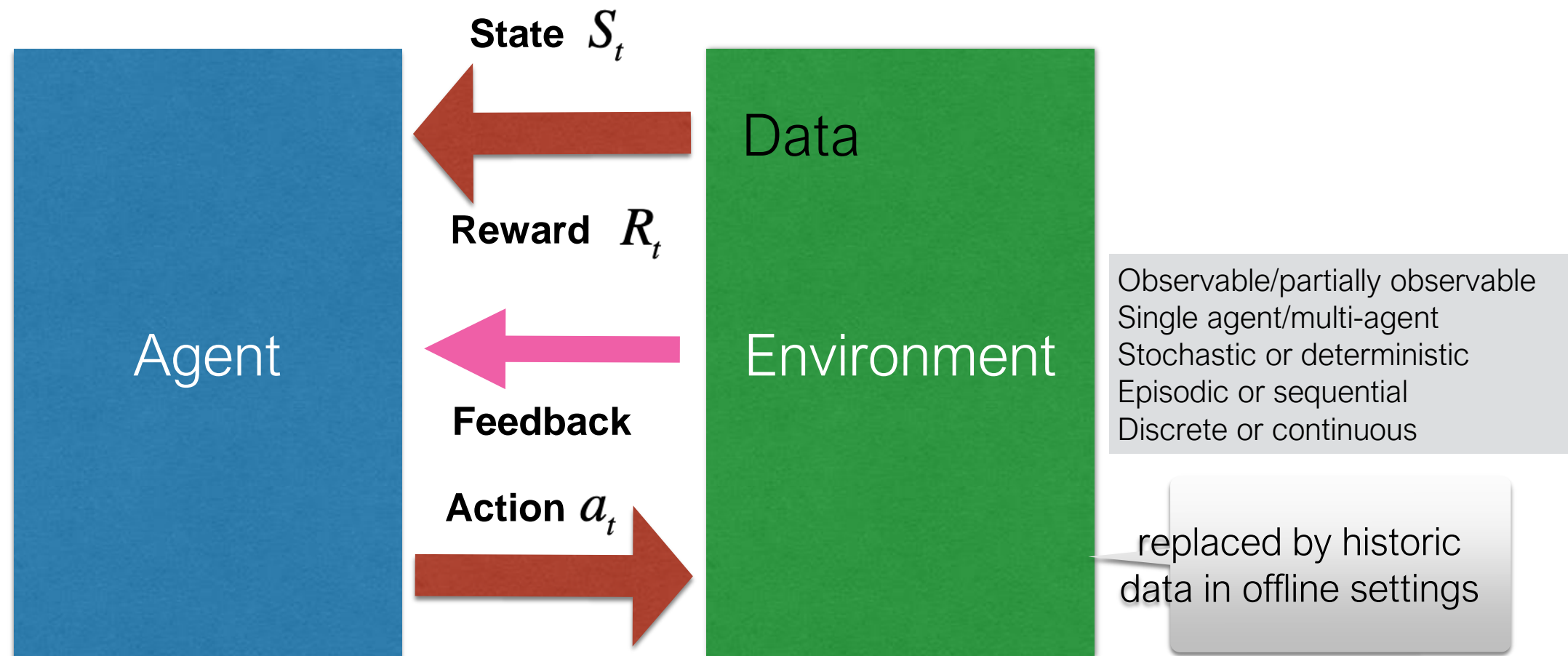
Why I like RL: 1) it implements Vapnik's principle, and 2) it incorporates a feedback loop

RL or IRL?



To learn like the humans, show we the robots use RL or IRL?

Reinforcement Learning



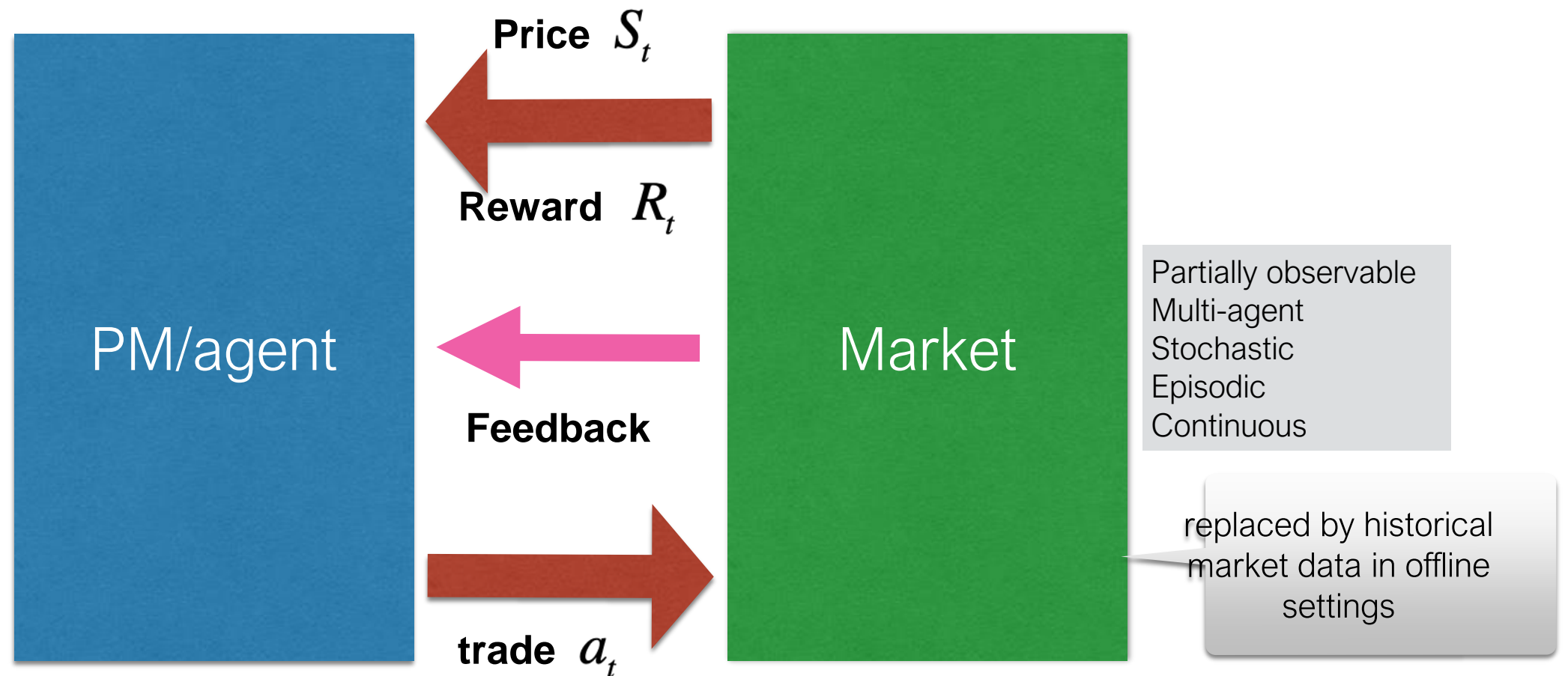
Reinforcement Learning (“**Action tasks**”): **sequential (multi-step) decision-making** by choosing multiple possible actions. As the state of the environment may change with time, RL involves planning and forecasting the future.

The objective of RL: **maximize the total reward** from taking actions

A **Feedback loop** is unique to RL, not encountered in SL or UL

RL tries to generalize methods of optimal control (Bellman’s dynamic programming) to work for real-world problems.

Portfolio management as a RL task

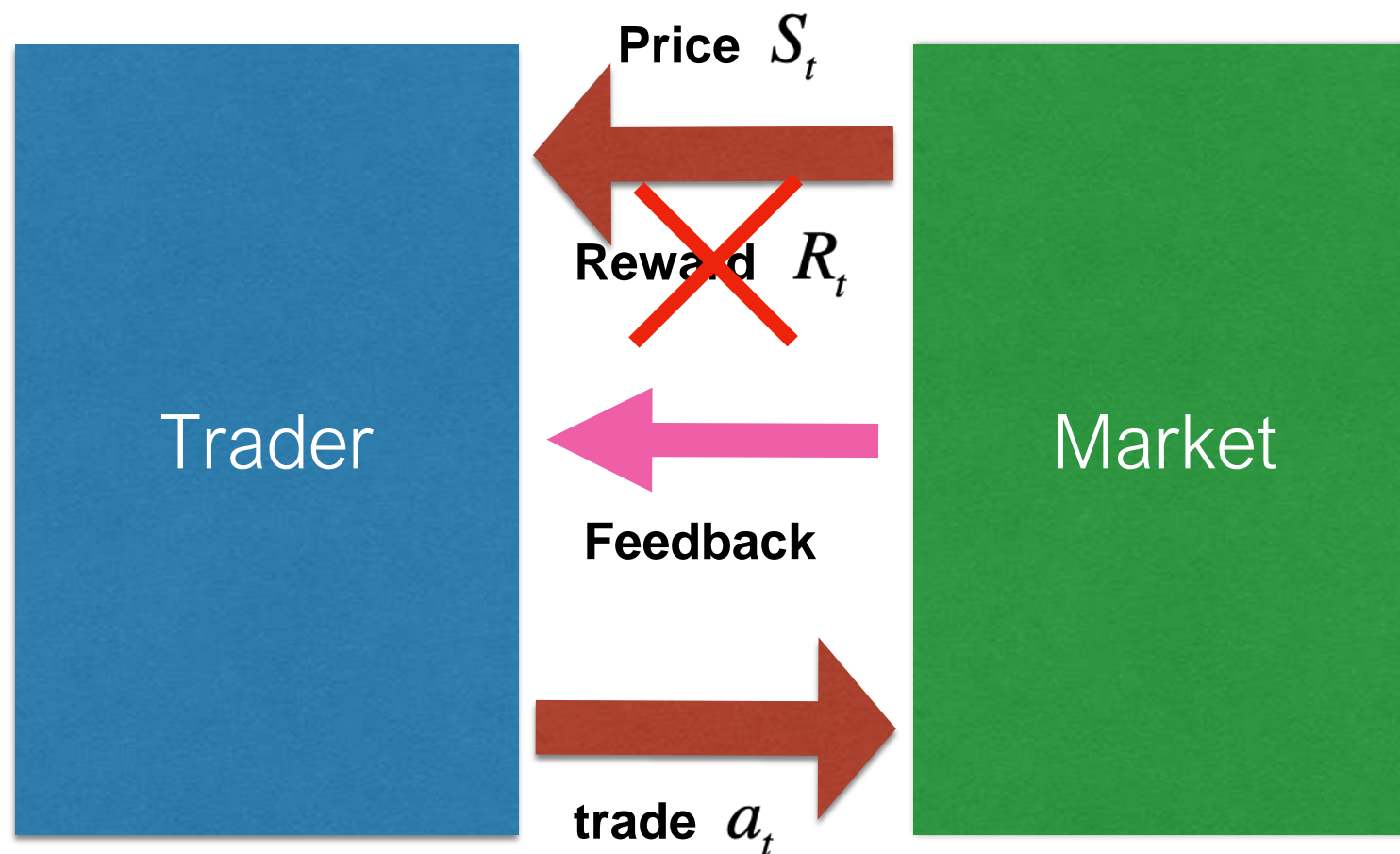


Portfolio management as RL: sequential (multi-step) decision-making by choosing multiple possible actions: buy/sell options and inject/withdraw cash. This involves involves planning and forecasting the future.

Feedback loop from trading: Large trades (or many accumulated and coherent small ones) make impact market prices.

Retirement optimization task: amounts to RL in a similar setting to the RL for PMs

IRL for portfolio optimization



IRL for portfolio trading: Given a history of sequential (multi-step) decision-making by an agent, the task is to learn the agent's **utility (reward)** function without observed rewards.

Applications: For a trader/trading desk, to infer (and hopefully improve) own strategy, where the rewards are not formalized as rewards of an MDP process.

Other potential applications: To infer the utility function of another agent. This assumes that actions (trades) are observable. More often, such proprietary data are not available! What can we do then?

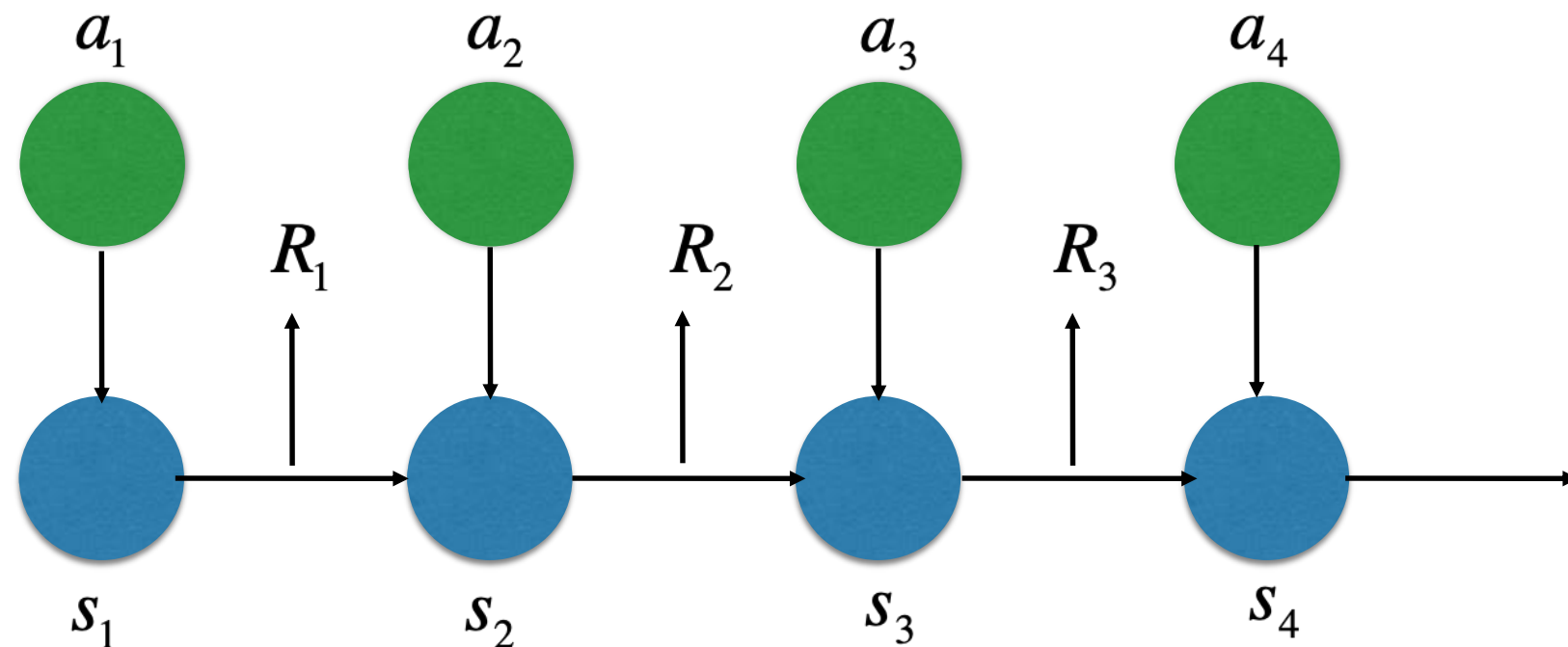
How to start with RL and IRL?

- The RL is a general data-driven (and often model-free) paradigm for problems of optimal control that aims at providing solutions for real-world problems of optimal control
- Most reported progress is in video games and robotics (“Deep RL”). These are typically problems with plenty of data, low noise level, and a relatively low (typically, in tens) dimensions of an action space. None of these is normally the case for applications for wealth or portfolio management.
- RL for finance is different from RL for Atari games - how should we proceed?
- Avoid the temptation to start with off-the-shelf Deep RL libraries
- Avoid the temptation to start with Deep Learning
- Start instead with simple models that avoid black-box architectures and admit semi-analytical solutions in terms of linear algebra or convex optimization
- Start with simple models of the agent reward
- Establish links with classical methods for optimal control.
- Move onto neural networks only 1) if needed, and 2) only after the previous steps are completed.

Markov Decision Processes

Markov Decision Processes:
be modulated by agent's actions

s_t the observable environment whose dynamics can
 a_t

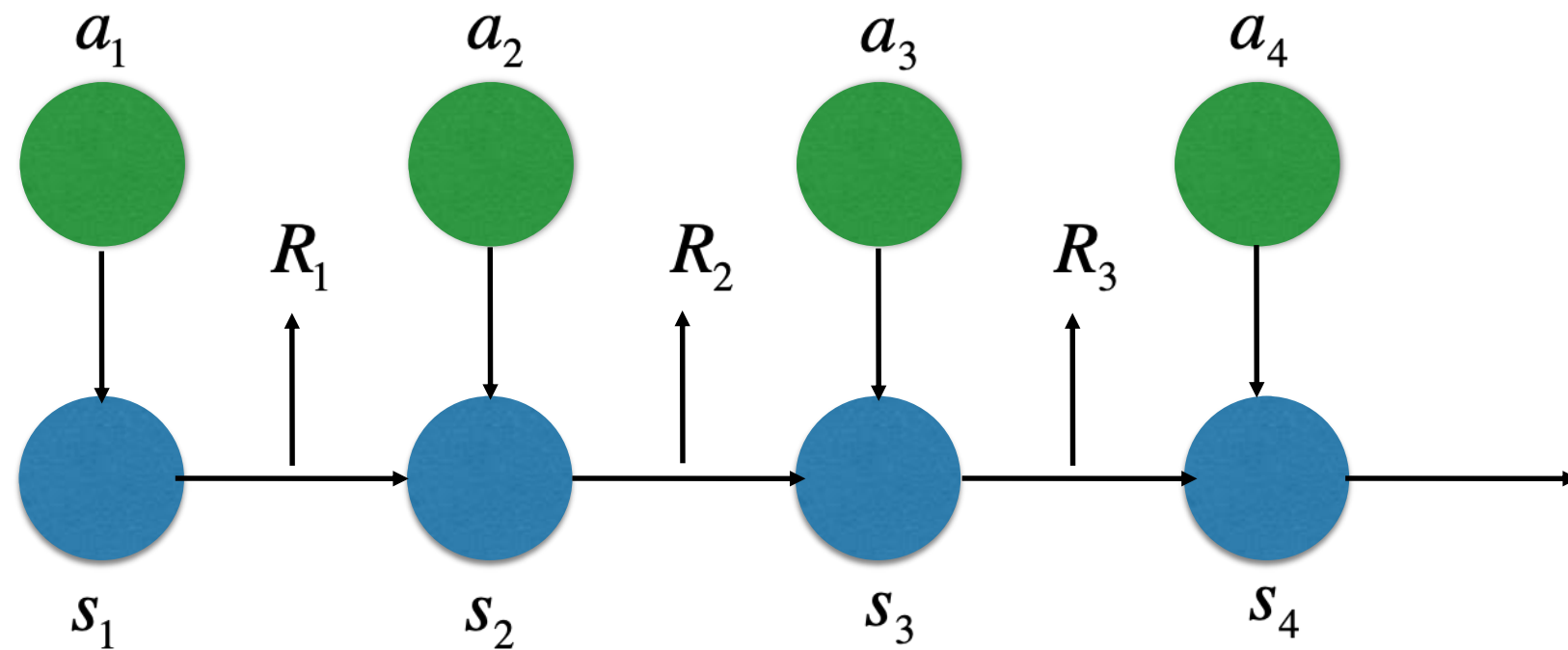


- $s_t \in S$: S is a set of **states** (discrete or continuous)
- $a_t \in A$: A is a set of **actions** (discrete or continuous)
- $p(s_{t+1} | s_t, a_t)$ are **transition probabilities**
- $R : S \times A \mapsto \mathbb{R}$ is a **reward function** (can depend on both state and action)
- $\gamma \in [0, 1]$ is a **discount factor**
- **Cumulative total reward**

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots = \sum_n \gamma^n R(s_n, a_n)$$

Decision policy

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots = \sum_n \gamma^n R(s_n, a_n)$$



- The **goal** in Reinforcement Learning is to **maximize the expected total reward**

$$\mathbb{E}[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots] = \mathbb{E}\left[\sum_n \gamma^n R(s_n, a_n)\right]$$

- This is achieved by an **optimal choice of a policy** $\pi : S \mapsto A$
- Whenever in state s_t we take action $a_t = \pi(s_t)$
- Policy π can be deterministic or stochastic (then $\pi(s_t)$ a probability distribution).

Bellman equation and Q-function

- The value function for policy π depends only on the current state, does not tell a RL agent what it should do

$$V_t^\pi(s) = \mathbb{E} \left[\sum_n \gamma^n R(s_n, a_n) \mid s_0 = s, \pi \right]$$

- The action-state value function (the Q-function): take action a , then follow π

$$Q_0^\pi(s, a) = \mathbb{E} \left[\sum_n \gamma^n R_n(s_n, a_n) \mid s_0 = s, a_0 = a, \pi \right] = R_0(s, a) + \mathbb{E} \left[\sum_{n=1} \gamma^n R_n(s_n, a_n) \mid \pi \right]$$

- The Bellman equation for the Q-value function

$$Q_t^\pi(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{s_{t+1}} \left[\gamma V_{t+1}^\pi(s_{t+1}) \right]$$

- The optimal Q-function:

$$Q_t^*(s_t, a_t) = \max_{\pi} Q_t^\pi(s_t, a_t)$$

- The optimal V-function:

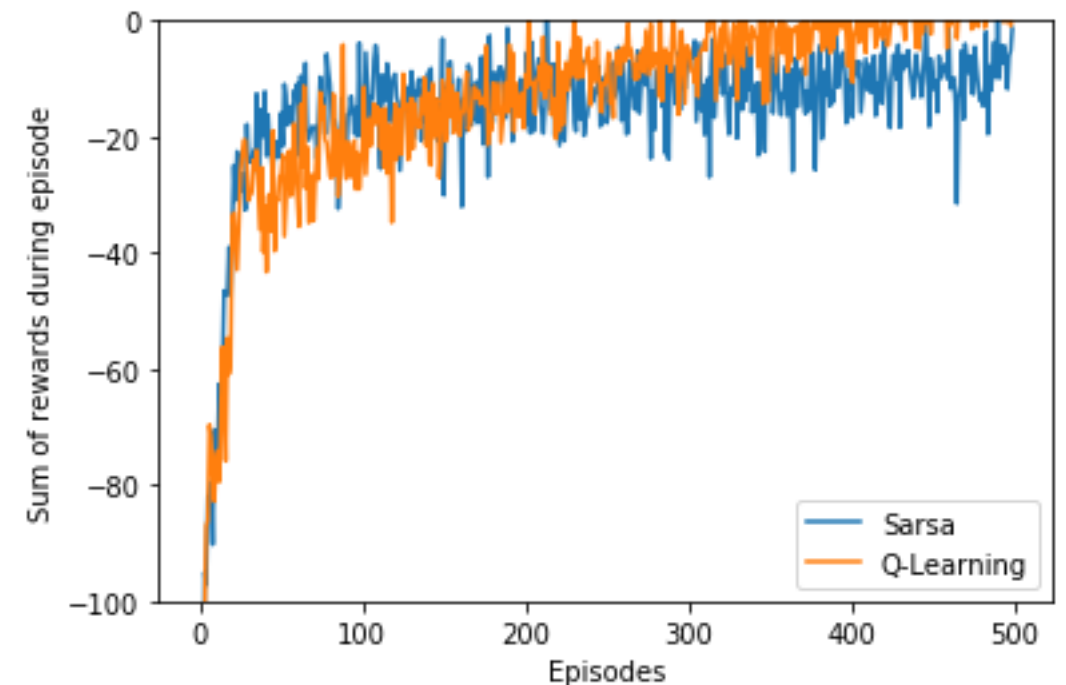
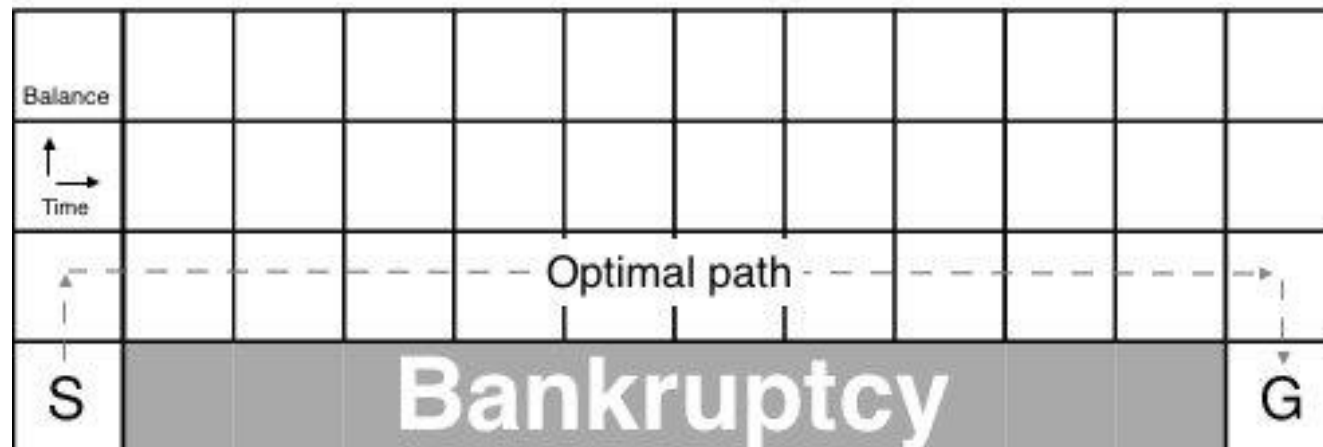
$$V_t^*(s_t) = \max_{a_t \in A} Q_t^*(s_t, a_t)$$

- The Bellman equation for the optimal Q-function

$$Q_t^*(s_t, a_t) = R_t(s_t, a_t) + \mathbb{E}_{s'} \left[\gamma V_{t+1}^*(s_{t+1}, a_t) \right]$$

$$Q_t^*(s_t, a_t) = R_t(s_t, a_t) + \max_{a_t \in A} \mathbb{E}_{s'} \left[\gamma Q_{t+1}^*(s_{t+1}, a_t) \right]$$

Financial cliff walking: the RL problem



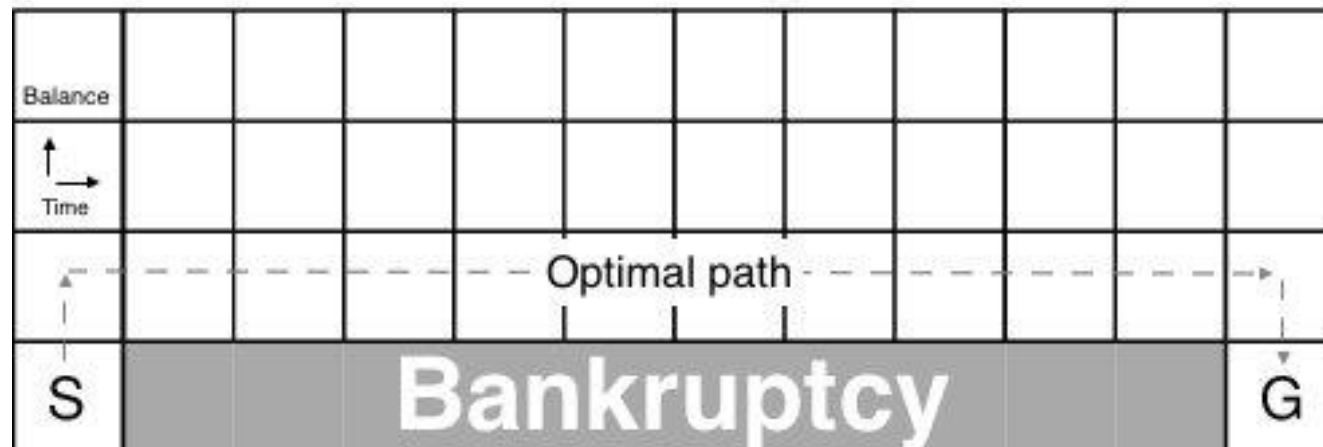
Financial cliff walking: (adaptation of the cliff walking problem from Sutton and Barto book): may be a simplest possible ‘financial’ problem for RL. The task is to maintain a minimum balance on a deposit account for a given period of time, then close the account. Transactions have costs, and there is a penalty if there is remaining balance at the final time.

This problem is very simple (no market randomness): Yet it takes about 300-400 episodes to learn not to overdraft the account!

RL for trading may not be easy!

Applications of RL: Optimal stock execution, option pricing (QLBS), ...

IRL for Financial cliff walking



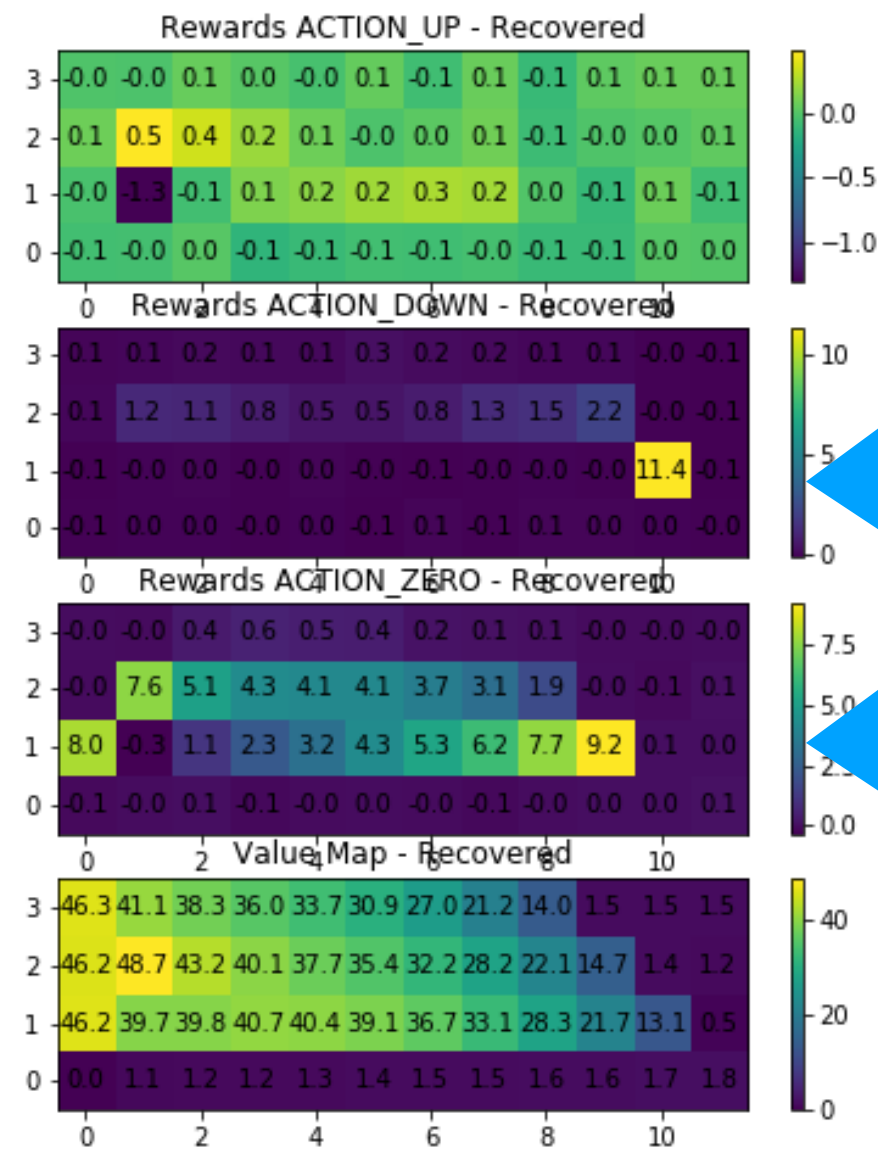
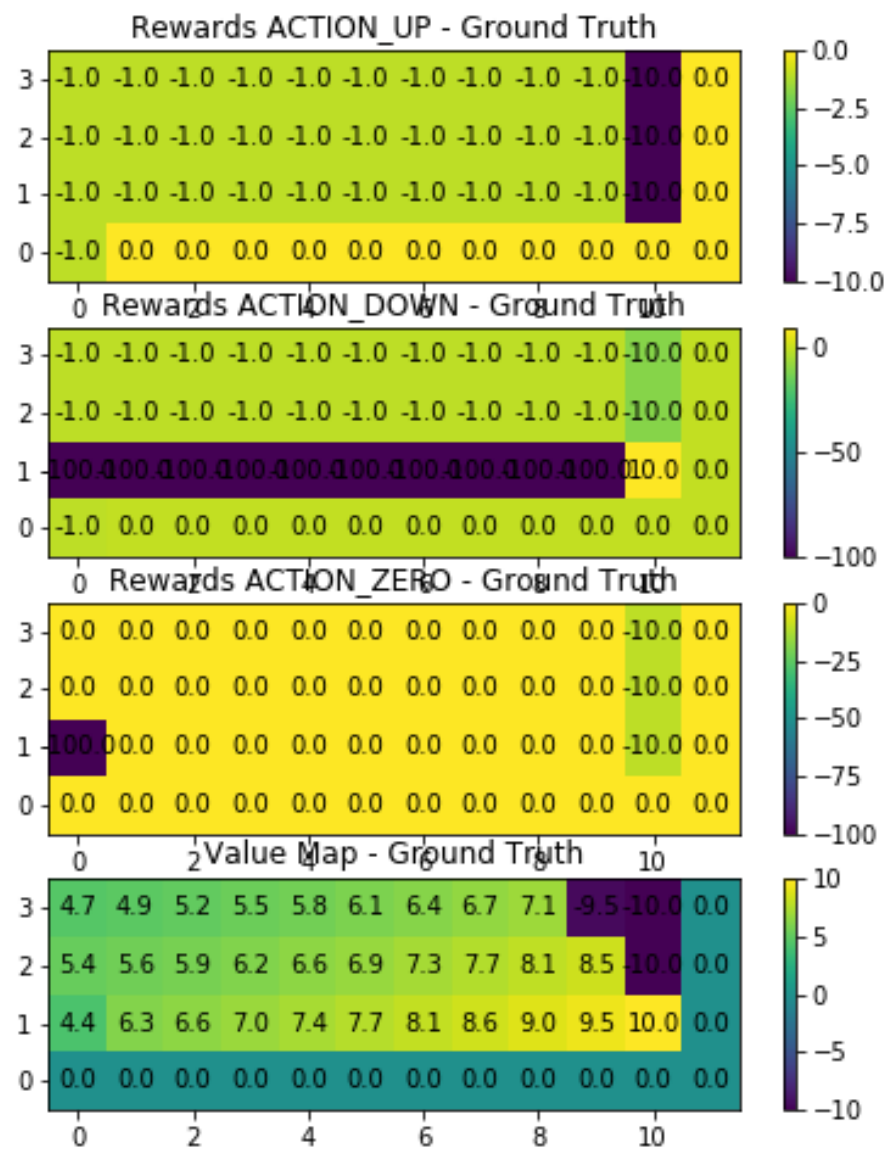
With IRL, we are given a history of states and actions, and the task is to infer the reward function from this data

Maximum Entropy (MaxEnt) IRL: data collected is assumed to correspond to an optimal stochastic policy. It assumes that sub-optimal actions are possible but infrequent encountered in data (and sampled according to the optimal policy)

IRL from Failure: One can learn not only from successful trajectories but also from failed ones

T-REX (TRajjectory-based EXtrapolation): learns the **intent** of the agent, does **not** assume optimality of demonstrations

MaxEnt IRL for FCW

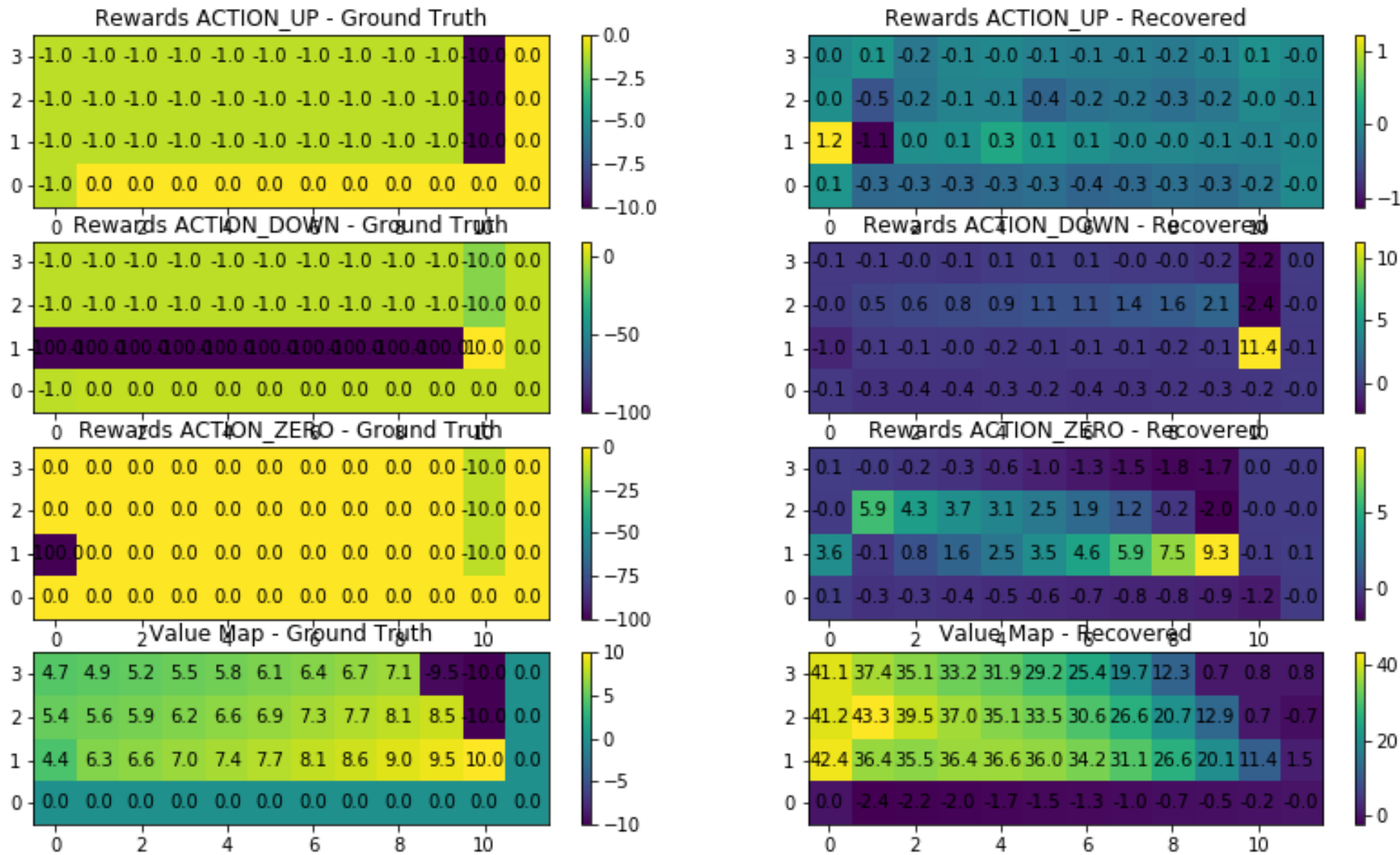


Captures a high reward of going down from this state

Assigns high rewards to these states because they are on the optimal path

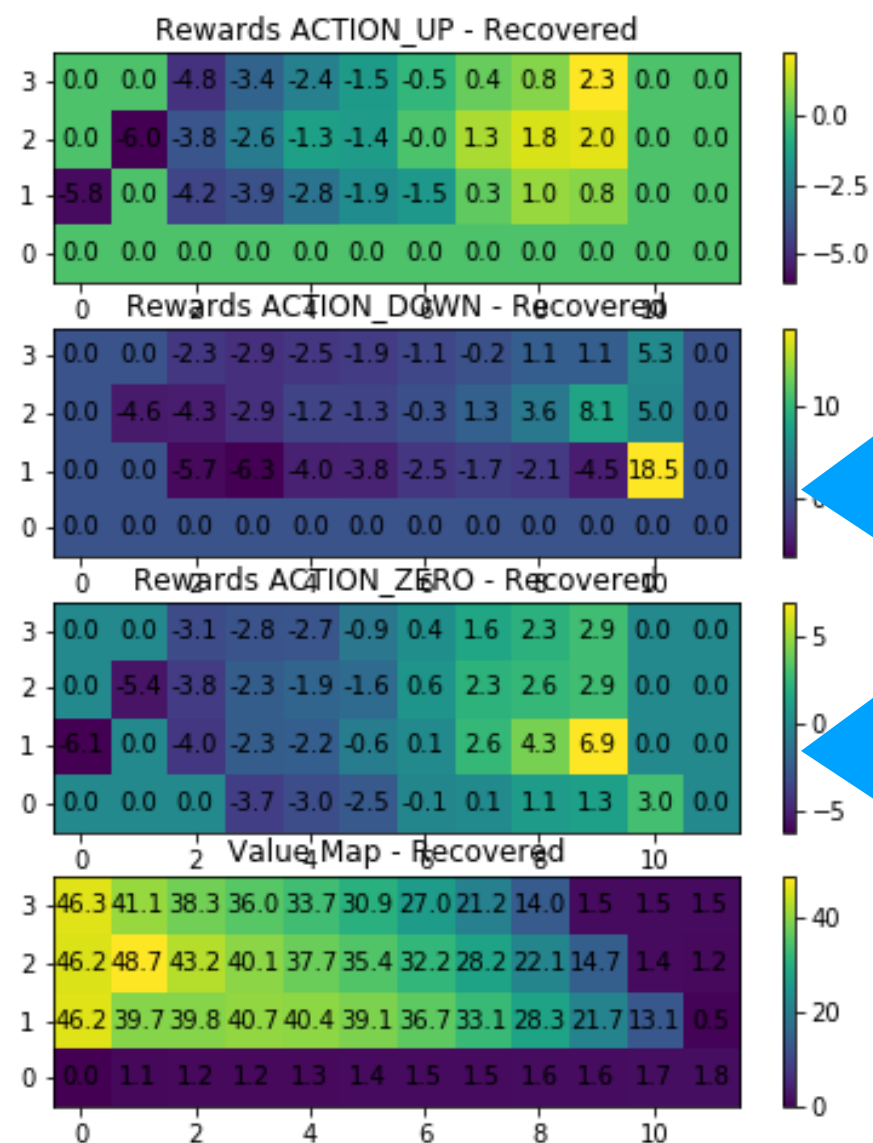
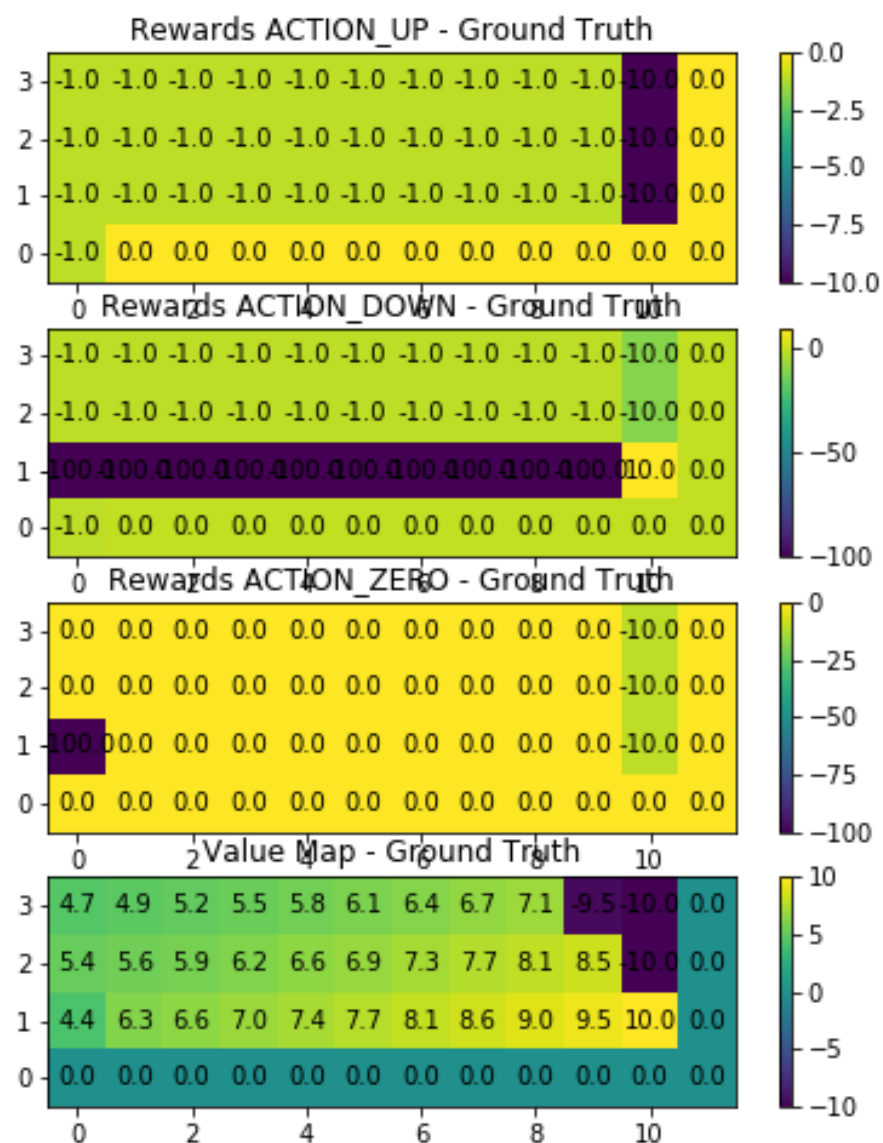
For the FCW problem, the MaxEnt IRL treats the bankruptcy events (moves down from the minimum deposit level) as occasional 'suboptimal' decisions

IRL from Failure for FCW



For the FCW problem, even though it is given both successful and failed trajectories, IRL from Failure produces a similar performance to MaxEnt IRL

IRL with T-REX



Captures a high reward of going down from this state

T-REX does **not** assign high rewards to these states just because they are on the optimal path

T-REX IRL performs better than MaxEnt IRL or IRL from Failure on the FCW example

Introduce G-learner and GIRL

- G-learner and GIRL are two related algorithms to perform, respectively, the direct RL and IRL for wealth management problems such as retirement plans.
- Any problem of portfolio optimization with possible cashflows at intermediate steps can be addressed with the G-learner algorithm, e.g. both cumulation and de-cumulation phases can be considered in the same framework.
- The **G-learner** algorithm offers an RL version of the goal-based approach to wealth management. It uses the RL approach called **G-learning** (which can be viewed a probabilistic extension of Q-learning)
- The (negative) reward in the G-learner algorithm is a penalty for under-performance relative to a running goal. The goal is a weighted combination (with weight α) of some external benchmark portfolio (e.g. an index) B and the current portfolio p growing with some factor β . In addition, the reward includes a quadratic penalty for under-performance of the next-step portfolio n relative to the running goal, with the risk aversion λ and trading cost parameter ω (**4 parameters of the quadratic reward function in total**)
- **GIRL** is an IRL algorithm that infers the reward parameters of a G-learning agent from the observed behavior including states and actions

Skip some math...

534

Using Eq.(770), the log-likelihood of observing data $\{X_t^{(k)}, a_t^{(k)}\}_{k=1}^N$ is (omitting a constant factor $-\frac{1}{2} \log(2\pi)$ in the second expression)

$$LL(\lambda) = \log \prod_{k=1}^N p_\lambda(a_t^{(k)} | X_t^{(k)}) = \sum_{k=1}^N \left(\frac{1}{2} \log c_2^{(k)}(\lambda) - \frac{c_2^{(k)}(\lambda)}{2} \left(a_t^{(k)} - \frac{c_1^{(k)}(\lambda)}{c_2^{(k)}(\lambda)} \right)^2 \right), \quad (771)$$

where $c_i^{(k)}(\lambda)$ with $i = 1, 2$ stands for expressions (769) evaluated on the k -th path. As this is a concave function of λ , its unique maximum can be easily found numerically using standard optimization packages.

Note that optimization in Eq.(771) refers to one particular value of t . This calculation can be repeated independently for different times t , producing a curve $\lambda_{impl}(t)$ that could be viewed as a term structure of implied risk aversion parameter.

To summarize this example, we see that when there is no market impact (a feedback loop in the system), MaxEnt IRL directly learns a one-step reward function. Parameters of this reward function can be estimated using the conventional Maximum Likelihood estimation with the MaxEnt stochastic policy.

10.3 IRL of a portfolio investor with G-learning

In Chapter 10, we introduced G-learning with quadratic rewards and Gaussian time-varying policies (GTVP) as a tool to optimize a dynamic asset portfolio. Such a model-based approach to G-learning amounts to a probabilistic version of the well-known Linear Quadratic Regulator. Thus far, we have considered two formulations that correspond, respectively, to self-financing and non-self-financing portfolios. While the first formulation is appropriate for modeling activities of asset managers, the second formulation with cash-flows at intermediate times is appropriate for tasks of wealth management and financial planning.

Here we shall take the second formulation of G-learning for an individual agent such as a retirement plan contributor or an individual brokerage account manager, and consider its inverse formulation. That is, we assume that we are given a history of dollar-nominated asset positions in an investment portfolio, jointly with a portfolio manager's decisions that include both injections or withdrawals of cash from the portfolio and asset allocation decisions. Additionally, we are given historical values of asset prices and expected asset returns for all assets in the investor universe. As a concrete example, we can consider a portfolio of stocks and a single bond, but the same formalism can be applied to other types of assets.

Recall that in Sect. 6.3 we obtained the stochastic policy (see Eq.(629))

$$\pi(\mathbf{u}_t | \mathbf{x}_t) = \pi_0(\mathbf{u}_t | \mathbf{x}_t) e^{\beta(G_t^\pi(\mathbf{x}_t, \mathbf{u}_t) - F_t^\pi(\mathbf{x}_t))}. \quad (772)$$

535

The quadratic reward function considered in Sect. 6.3 corresponds to the quadratic action-value function (see Eqs.(627) and (622))

$$\begin{aligned} F_t^\pi(\mathbf{x}_t) &= \mathbf{x}_t^T \mathbf{F}_t^{(xx)} \mathbf{x}_t + \mathbf{x}_t^T \mathbf{F}_t^{(x)} + F_t^{(0)} \\ G_t^\pi(\mathbf{x}_t, \mathbf{u}_t) &= \mathbf{x}_t^T \mathbf{Q}_t^{(xx)} \mathbf{x}_t + \mathbf{x}_t^T \mathbf{Q}_t^{(xu)} \mathbf{u}_t + \mathbf{u}_t^T \mathbf{Q}_t^{(ux)} \mathbf{x}_t + \mathbf{u}_t^T \mathbf{Q}_t^{(uu)} \mathbf{u}_t + \mathbf{x}_t^T \mathbf{Q}_t^{(x)} + \mathbf{u}_t^T \mathbf{Q}_t^{(u)} + Q_t^{(0)}, \end{aligned} \quad (773)$$

where

$$\begin{aligned} \mathbf{Q}_t^{(xx)} &= -\lambda \hat{\Sigma}_t + \gamma (\mathbf{A}_t^T \bar{\mathbf{F}}_{t+1}^{(xx)} \mathbf{A}_t + \tilde{\Sigma}_r \circ \bar{\mathbf{F}}_{t+1}^{(xx)}) \\ \mathbf{Q}_t^{(xu)} &= 2\mathbf{Q}_t^{(xx)} \\ \mathbf{Q}_t^{(uu)} &= \mathbf{Q}_t^{(xx)} - \Omega \\ \mathbf{Q}_t^{(x)} &= 2\lambda \hat{P}_{t+1}(1 + \bar{\mathbf{r}}_t) + \gamma \mathbf{A}_t^T \bar{\mathbf{F}}_{t+1}^{(x)} \\ \mathbf{Q}_t^{(u)} &= \mathbf{Q}_t^{(x)} - \mathbb{1} \\ Q_t^{(0)} &= -\lambda \hat{P}_{t+1}^2 + \gamma F_{t+1}^{(0)}. \end{aligned} \quad (774)$$

Assume that we have historical data that includes a set of D trajectories ζ_i where $i = 1, \dots, D$ of state-action pairs $(\mathbf{x}_t, \mathbf{a}_t)$ where trajectory i starts at some time t_{0i} and runs until time T_i . Consider a single trajectory ζ from this collection, and set for this trajectory the start time $t = 0$ and the end time T . As individual trajectories are considered independent, they will enter additively in the final log-likelihood of the problem. We assume that dynamics are Markovian in the pair $(\mathbf{x}_t, \mathbf{u}_t)$, with a generative model $p_\theta(\mathbf{x}_{t+1}, \mathbf{u}_t | \mathbf{x}_t) = \pi_\theta(\mathbf{u}_t | \mathbf{x}_t) p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t)$ where Θ stands for a vector of model parameters. The probability of observing trajectory ζ is given by the following expression

$$P(\mathbf{x}, \mathbf{u} | \Theta) = p_\theta(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi_\theta(\mathbf{u}_t | \mathbf{x}_t) p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t). \quad (775)$$

Here $p(\mathbf{x}_0)$ is a marginal probability of \mathbf{x}_t at the start of the i -th demonstration. Assuming that the initial values \mathbf{x}_0 are fixed, this gives the following log-likelihood for data $\{\mathbf{x}_t, \mathbf{a}_t\}_{t=0}^T$ observed for trajectory ζ :

$$LL(\theta) := \log P(\mathbf{x}, \mathbf{u} | \Theta) = \sum_{t \in \zeta} (\log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t) + \log p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t)). \quad (776)$$

Transition probabilities $p_\theta(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t)$ entering this expression can be obtained from the state equation

$$\mathbf{x}_{t+1} = \mathbf{A}_t(\mathbf{x}_t + \mathbf{u}_t) + (\mathbf{x}_t + \mathbf{u}_t) \circ \tilde{\varepsilon}_t, \quad \mathbf{A}_t := \text{diag}(1 + \bar{\mathbf{r}}_t), \quad \tilde{\varepsilon}_t := (0, \varepsilon_t), \quad (777)$$

where ε_t is a Gaussian noise with covariance Σ_r (see Eq.(617)). Writing $\mathbf{x}_t = (x_t^{(0)}, \mathbf{x}_t^{(r)})$ where $x_t^{(0)}$ is the value of a bond position and $\mathbf{x}_t^{(r)}$ are the values of positions in risky assets, and similarly for \mathbf{u}_t and \mathbf{A}_t , this produces

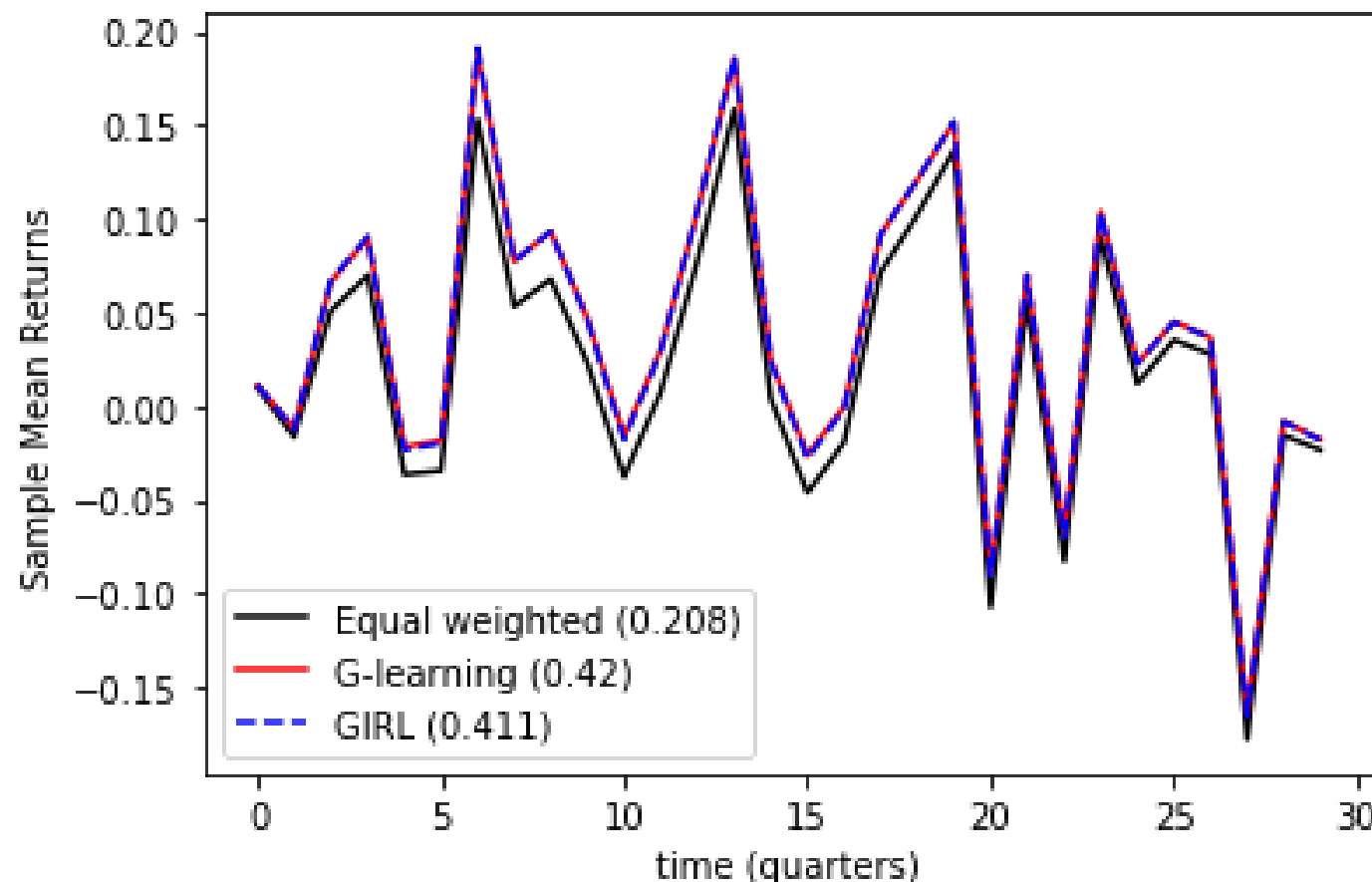
Experiments with the G-learner

- The objective is to demonstrate that efficient portfolio management can be achieved with a large number of assets and for long horizons. We choose 100 assets (99 stocks and a bond) and 30 quarterly time steps for numerical illustrations
- The G-learner bases its actions on the outputs of an ‘alpha-model’ (a model for expected equity returns)
- We use simulations with some ad-hoc model parameters (picked to produce realistically-looking optimal installments), and with an alpha-model producing expected returns that are weakly correlated with realized returns:



Results for the G-learner and GIRL

- The G-learner is able to harvest weak signals from the ‘alpha-model’ **to boost the Sharpe ratio by 0.21** relatively to a fixed equally-weighted portfolio that ignores the signals and keeps the portfolio fixed over its lifetime.
- Once demonstrations are produced by the G-learner with fixed parameters, the IRL algorithm (GIRL) manages to infer these parameters, and hence mimic the performance of the G-learner:



G-learner and GIRL work together

- How G-learner and GIRL can work together:
- Real investors are modeled as G-learning agents
- GIRL is used to infer the reward function parameters
- Inferred parameters can be used to cluster investors in classes defined by their saving/investment behavior - meaningful clusters by construction
- GIRL can be used for robo-advising



- Reference.
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3543852

IRL for the market as a whole

- The G-learner and GIRL are algorithms to do IRL for individual investors. Another way to use IRL is to apply it to a market-wide agent that embodies collective dynamics of **all** market participants. This is the IRL approach to an **Invisible Hand** of the market.
- A bounded-rational IRL agent has a Markowitz reward function regularized by entropy



- Reference: I.Halperin and I.Feldshteyn,
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3174498

Back to the book:

- Chapt. 9 (intro to RL) -> Chapt. 10 (applications of RL to trading) -> Chapt. 11 (IRL)
- RL: applications to portfolio optimization, wealth management, option pricing
- IRL extensions: improving on an expert (e.g. the T-REX algorithm), the IRL Invisible Hand model, etc.
- Chapter 12 (research topics):
 - ◆ use insights from physics to generalize market dynamics suggested by IRL
 - ◆ Physics-inspired methods for machine learning (non-equilibrium models, tensor networks etc.)
 - ◆ Physics envy: how to do a Grand Unification for machine learning?

