Reinforcement Learning the Learning Rate in Probabilistic Reversal Learning Tasks

Semester Project

Simon Christen

Student D-ITET, ETHZ

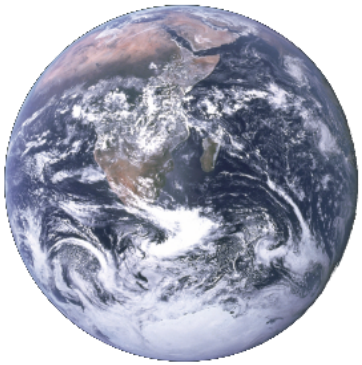
Supervisor: Dr. Kerstin Preuschoff

Department of Economics, University of Zürich

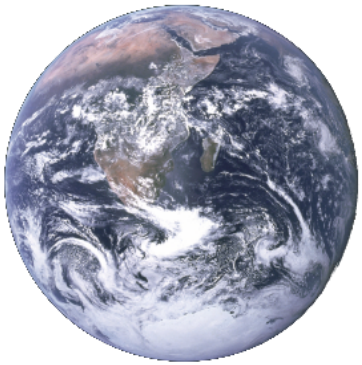
May 25, 2011

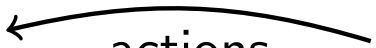
Outline

Motivation and Question

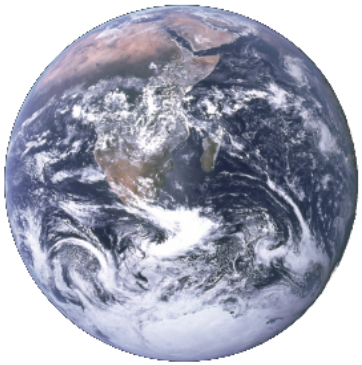


Motivation and Question





Motivation and Question

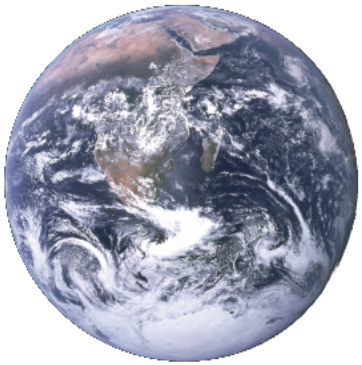


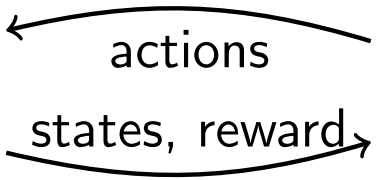
actions

states, reward

Motivation and Question

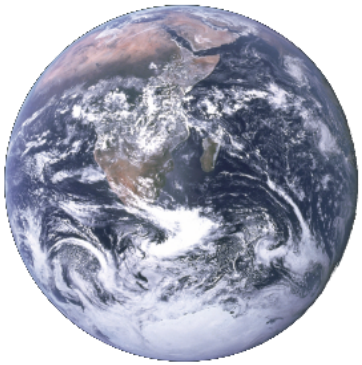
- Truth* changes
  $\longrightarrow$ Volatility

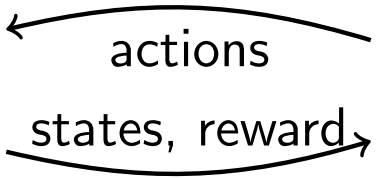




* Action of best reward, $Q^*(s, a)$, best stimuli.

Motivation and Question

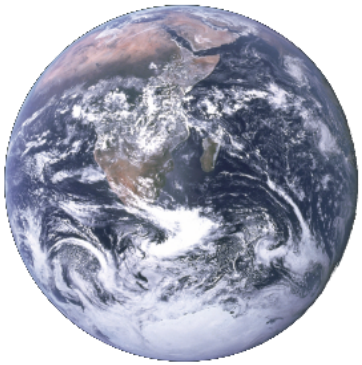


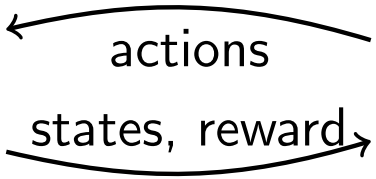


- Truth* changes
  $\longrightarrow$ Volatility
- Observation* is noisy
  $\longrightarrow$ Risk

* Action of best reward, $Q^*(s, a)$, best stimuli. ** States, Reward

Motivation and Question

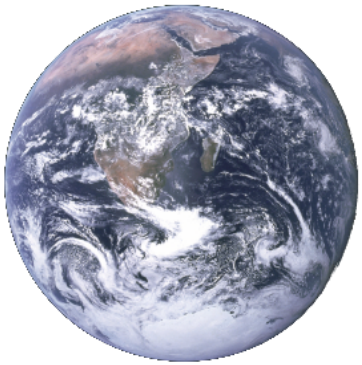


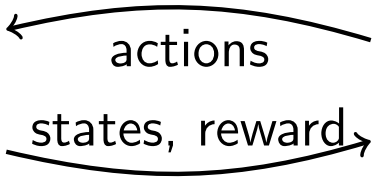


- Truth* changes
  $\longrightarrow$ Volatility
- Observation* is noisy
  $\longrightarrow$ Risk
- How conservative do I change my estimate of the truth?
  $\longrightarrow$ Learning rate λ

* Action of best reward, $Q^*(s, a)$, best stimuli. ** States, Reward

Motivation and Question



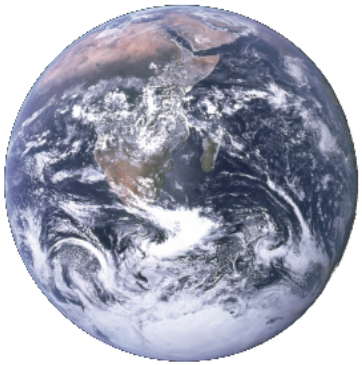


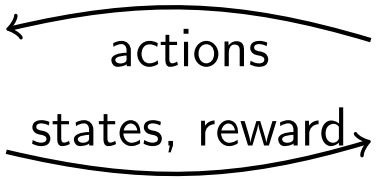
- Truth* changes
  $\longrightarrow$ Volatility
- Observation* is noisy
  $\longrightarrow$ Risk
- How conservative do I change my estimate of the truth?
  $\longrightarrow$ Learning rate λ

Background The supervisor proposed a new model free learning rate control method. The target problem is probabilistic reversal learning.

* Action of best reward, $Q^*(s, a)$, best stimuli. ** States, Reward

Motivation and Question





- Truth* changes
  $\longrightarrow$ Volatility
- Observation* is noisy
  $\longrightarrow$ Risk
- How conservative do I change my estimate of the truth?
  $\longrightarrow$ Learning rate λ

Background The supervisor proposed a new model free learning rate control method. The target problem is probabilistic reversal learning.

Task Explore the suggested algorithm for different parameters and compare to other methods. This involves parameters for the Q-learning as for the tabu policy learning method.

* Action of best reward, $Q^*(s, a)$, best stimuli. ** States, Reward

Probabilistic Reversal Learning

Real Experiment

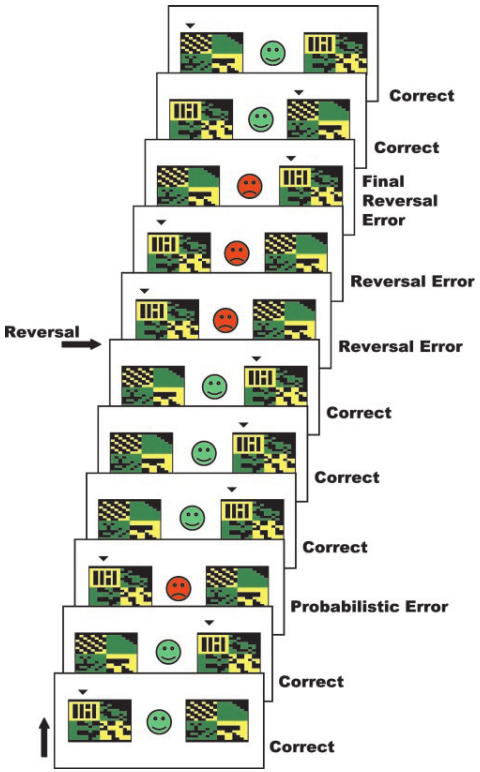


Figure: Cools et al. 2002

Probabilistic Reversal Learning

Real Experiment

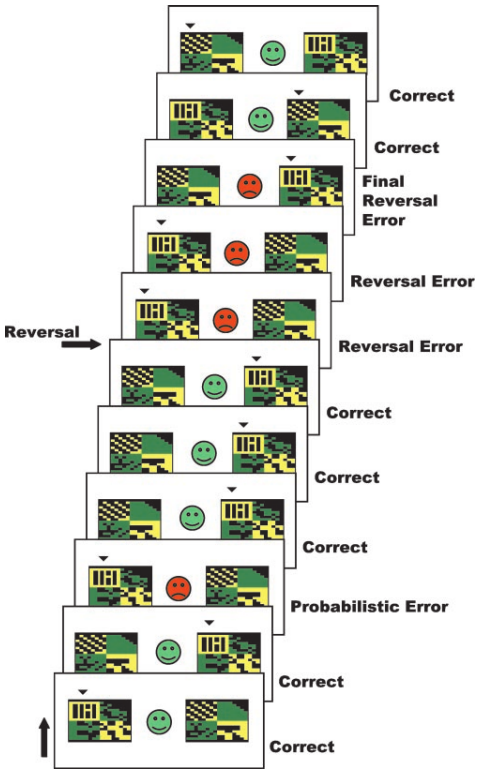


Figure: Cools et al. 2002

Model

Probabilistic Reversal Learning

Real Experiment

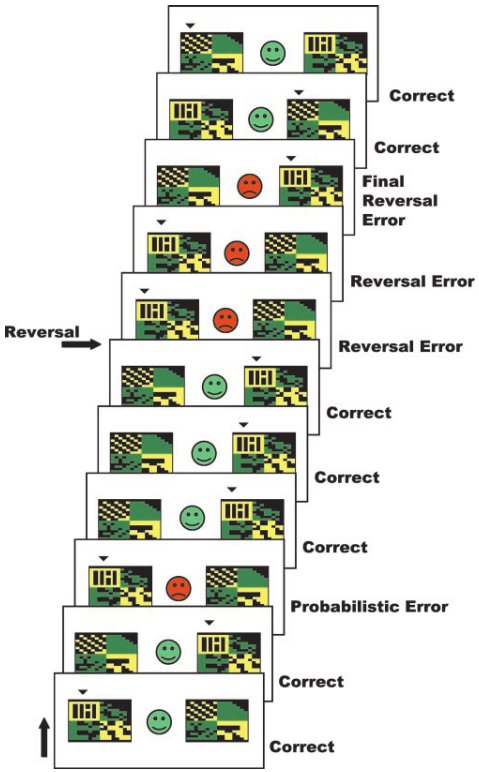


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Probabilistic Reversal Learning

Real Experiment

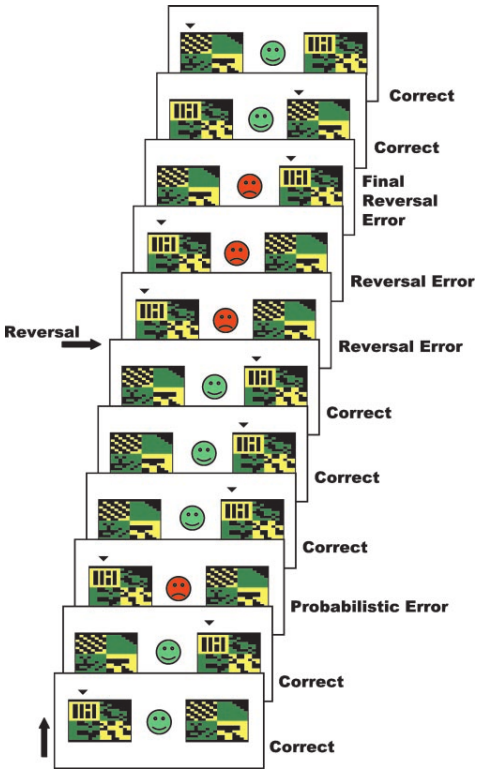


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Probabilistic Reversal Learning

Real Experiment

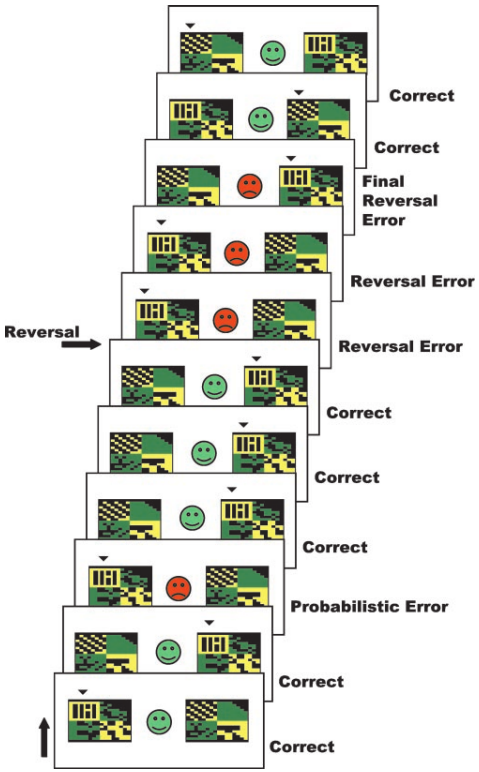


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Probabilistic Reversal Learning

Real Experiment

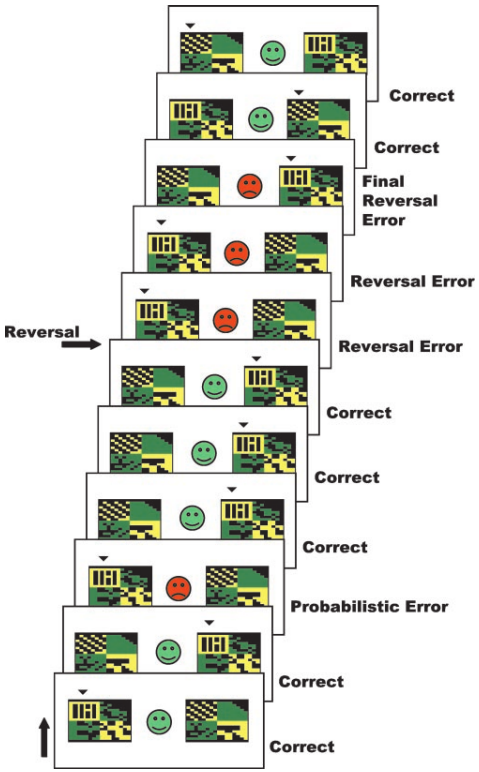


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Probabilistic Reversal Learning

Real Experiment

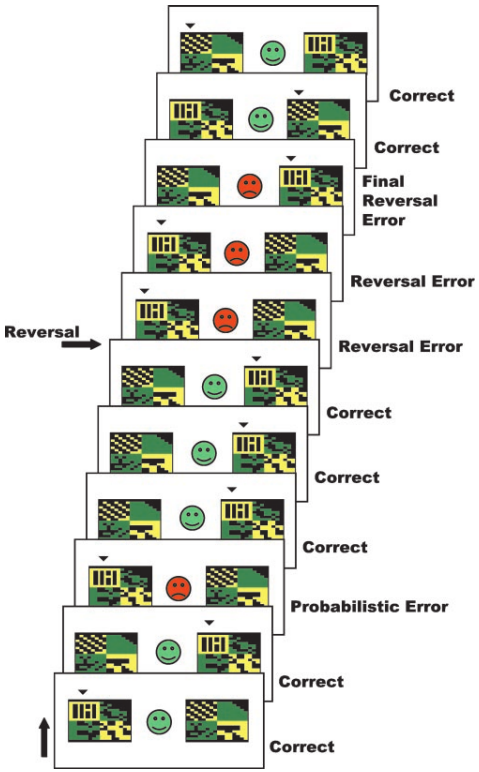


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Probabilistic Reversal Learning

Real Experiment

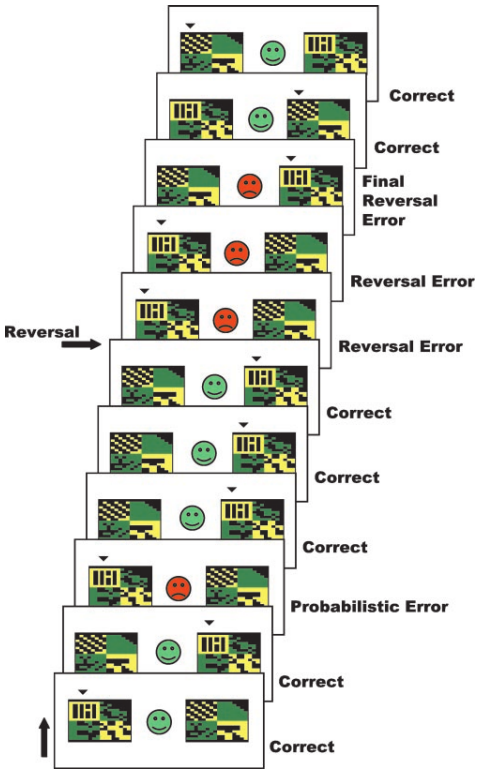


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Probabilistic Reversal Learning

Real Experiment

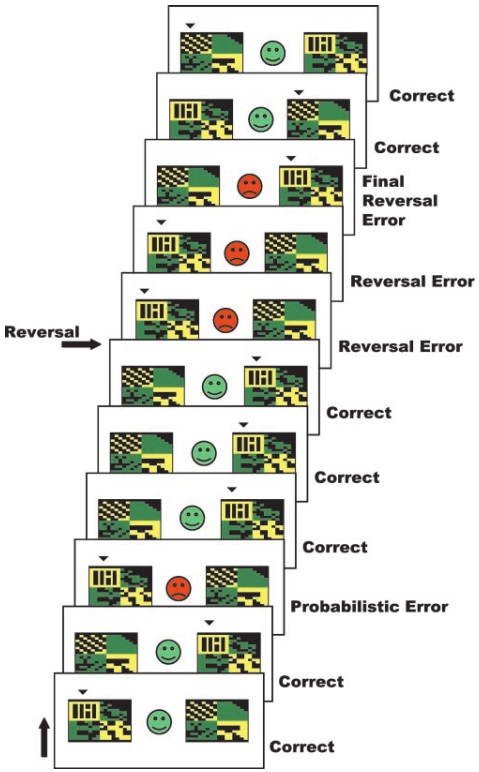


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Probabilistic Reversal Learning

Real Experiment

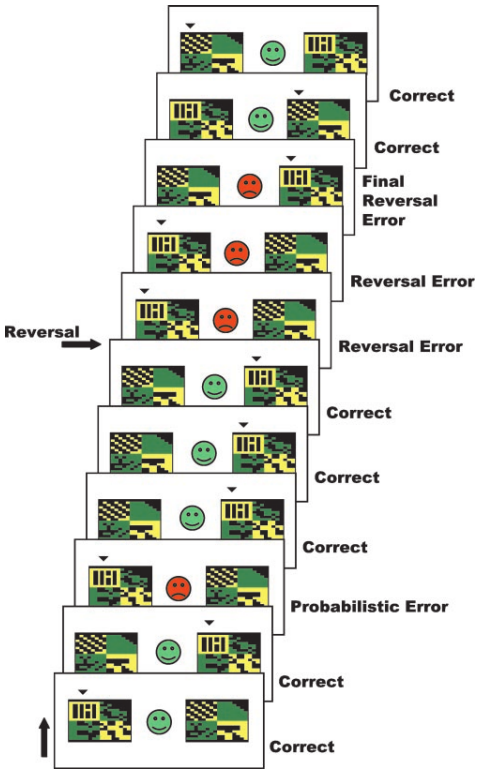


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

Env Sets better stimuli: $u_t \in \{0, 1\}$

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Probabilistic Reversal Learning

Real Experiment

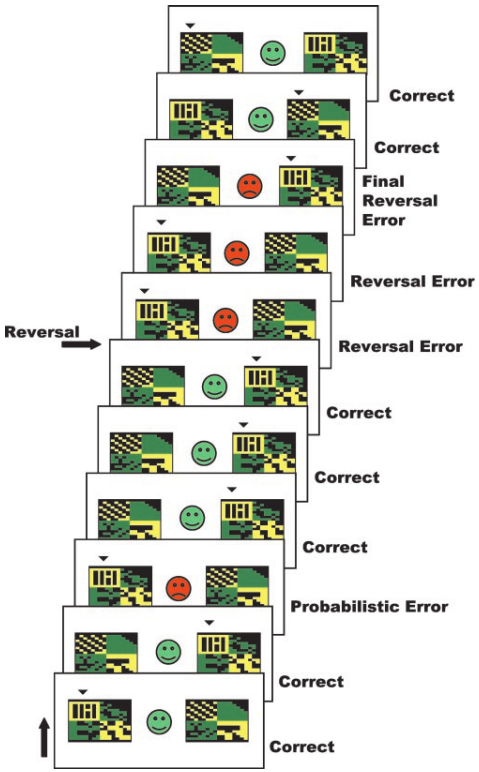


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

Env Sets better stimuli: $u_t \in \{0, 1\}$

Env Generate

$$x_t = \begin{cases} p & \text{if } u_t = 1 \\ 1 - p & \text{if } u_t = 0 \end{cases}$$

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Probabilistic Reversal Learning

Real Experiment

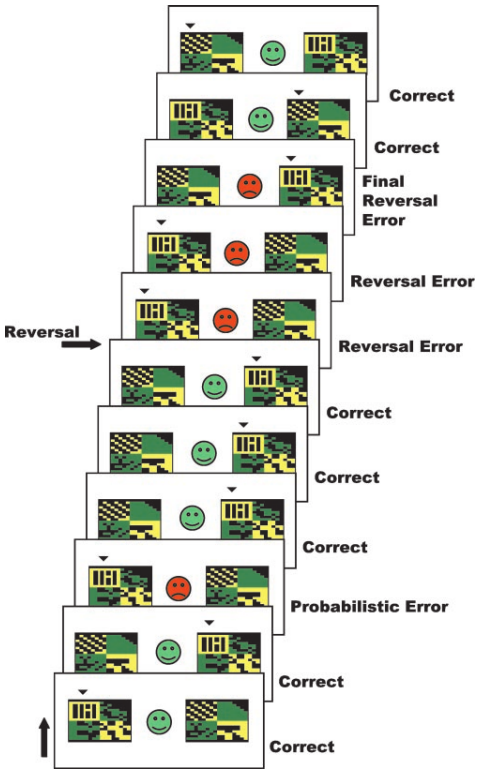


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

Env Sets better stimuli: $u_t \in \{0, 1\}$

Env Generate

$$x_t = \begin{cases} p & \text{if } u_t = 1 \\ 1 - p & \text{if } u_t = 0 \end{cases}$$

Env Reveals

$$\mathbf{y}_t \sim \text{Bern}(x_t)$$

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Probabilistic Reversal Learning

Real Experiment

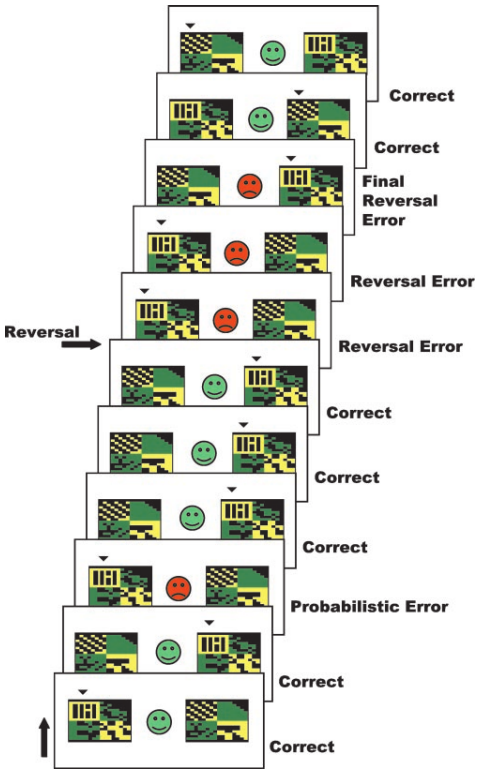


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

Env Sets better stimuli: $u_t \in \{0, 1\}$

Env Generate

$$x_t = \begin{cases} p & \text{if } u_t = 1 \\ 1 - p & \text{if } u_t = 0 \end{cases}$$

Env Reveals

$$\mathbf{y}_t \sim \text{Bern}(x_t)$$

Ag Estimates x_t by discounted recursive mean (ML):

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Probabilistic Reversal Learning

Real Experiment

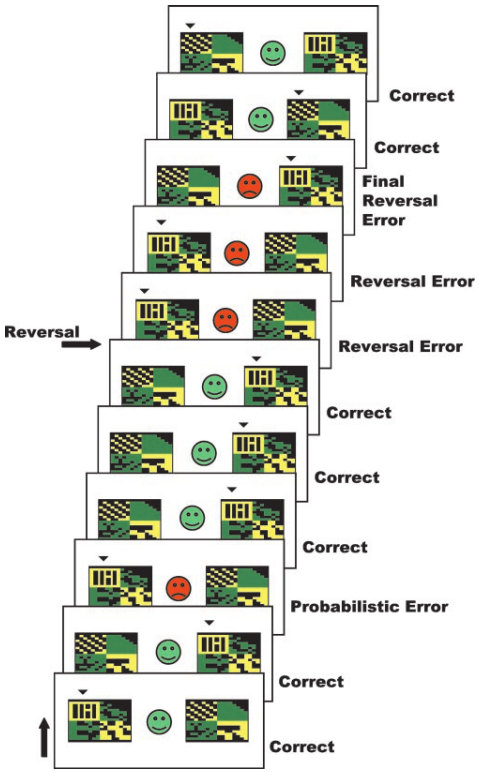


Figure: Cools et al. 2002

Model

Env Sets the better stimuli $u_t \in \{r, l\}$

Ag takes action $\mathbf{a}_t \in \{r, l\}$

Env Generates reward:

$$\Pr\{\text{reward} \mid \mathbf{a}_t = u_t\} = p$$
$$\Pr\{\text{reward} \mid \mathbf{a}_t \neq u_t\} = 1 - p$$

Ag detects better stimuli:

action	reward	better stimuli $\mathbf{y}_t$
l	yes	1
l	no	0
r	yes	0
r	no	1

Ag trend of the better stimuli by discounted recursive mean:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag estimate of better stimuli by

$$\mathbf{a}_{t+1} = \hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

Our Experiment

1. code left/right $\longrightarrow 1/0$
2. kick the action-reward thing*

Env Sets better stimuli: $u_t \in \{0, 1\}$

Env Generate

$$x_t = \begin{cases} p & \text{if } u_t = 1 \\ 1 - p & \text{if } u_t = 0 \end{cases}$$

Env Reveals

$$\mathbf{y}_t \sim \text{Bern}(x_t)$$

Ag Estimates x_t by discounted recursive mean (ML):

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + \lambda \left(\mathbf{y}_t - \hat{\mathbf{x}}_{t-1}\right)$$

Ag Estimates u_t:

$$\hat{\mathbf{u}}_t = \begin{cases} l & \text{if } \hat{\mathbf{x}} \geq 0.5 \\ r & \text{else} \end{cases}$$

* Instead of having a two complementary action model, we reduce it to a one action model. The action is given and $\mathbf{y}_t$ can be considered in a reinforcement learning context as the reward.

Reinforcement Learning

The Agent - Environment Setup

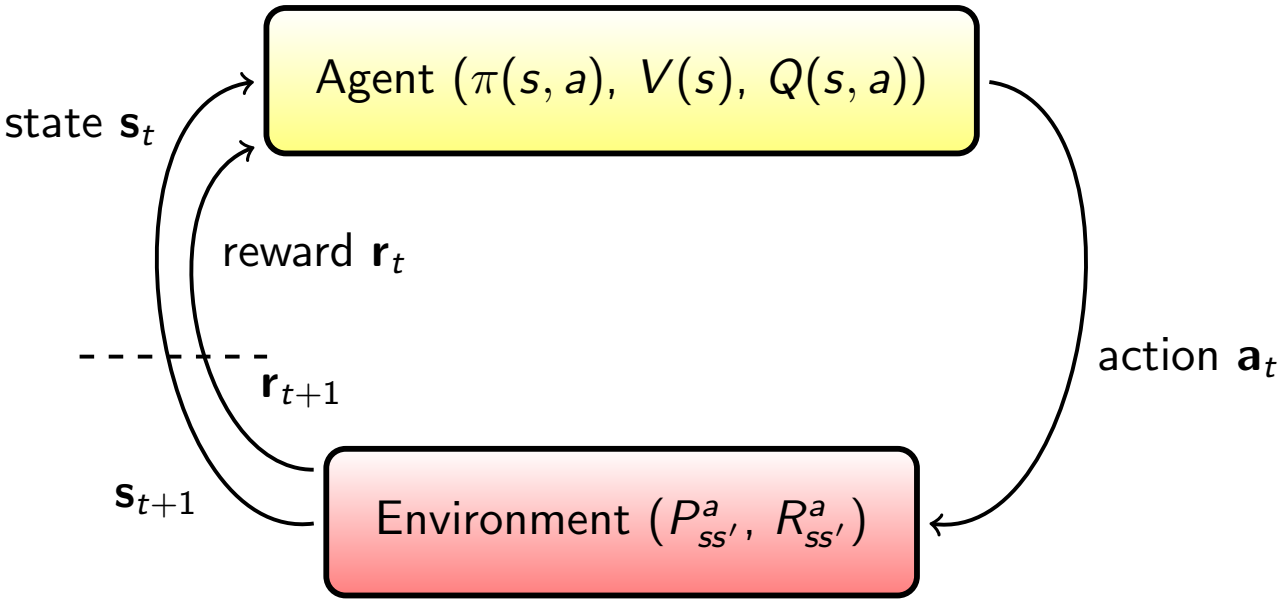
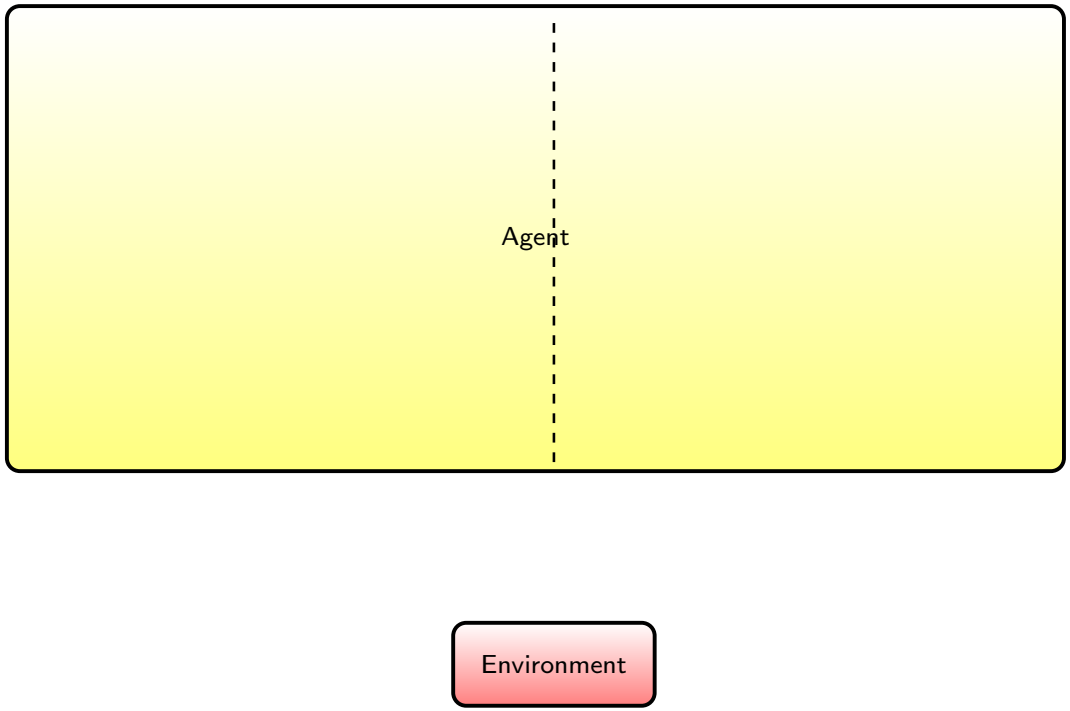


Figure: Reinforcement Learning Setup

Reinforcement Learning

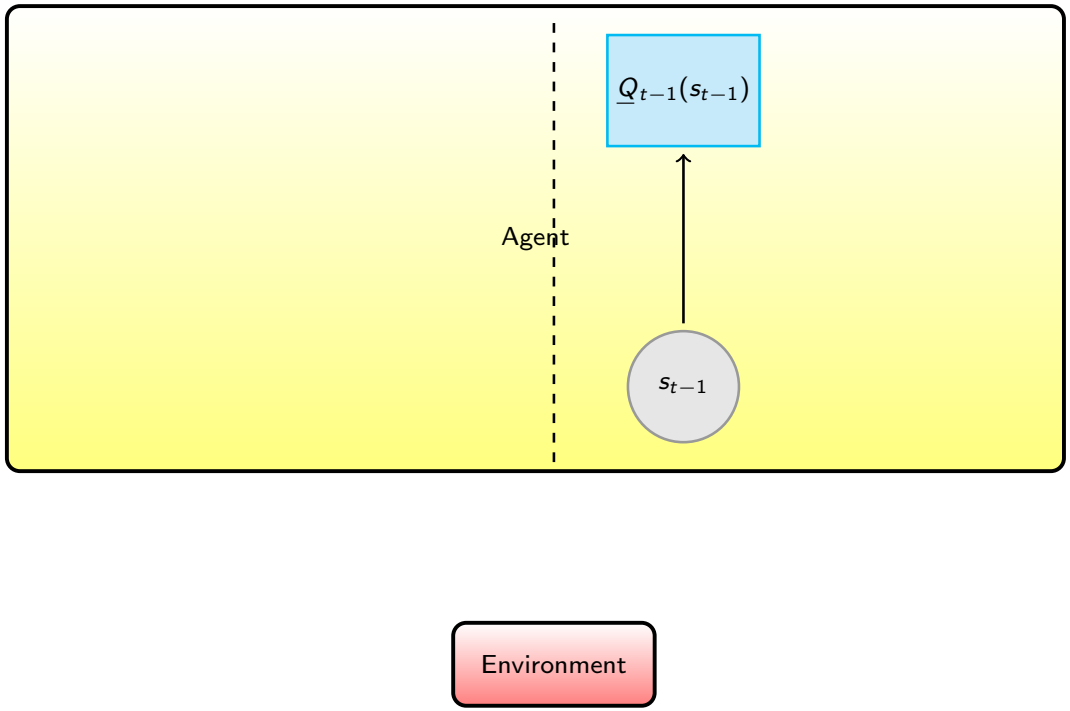
Q-Learning



*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

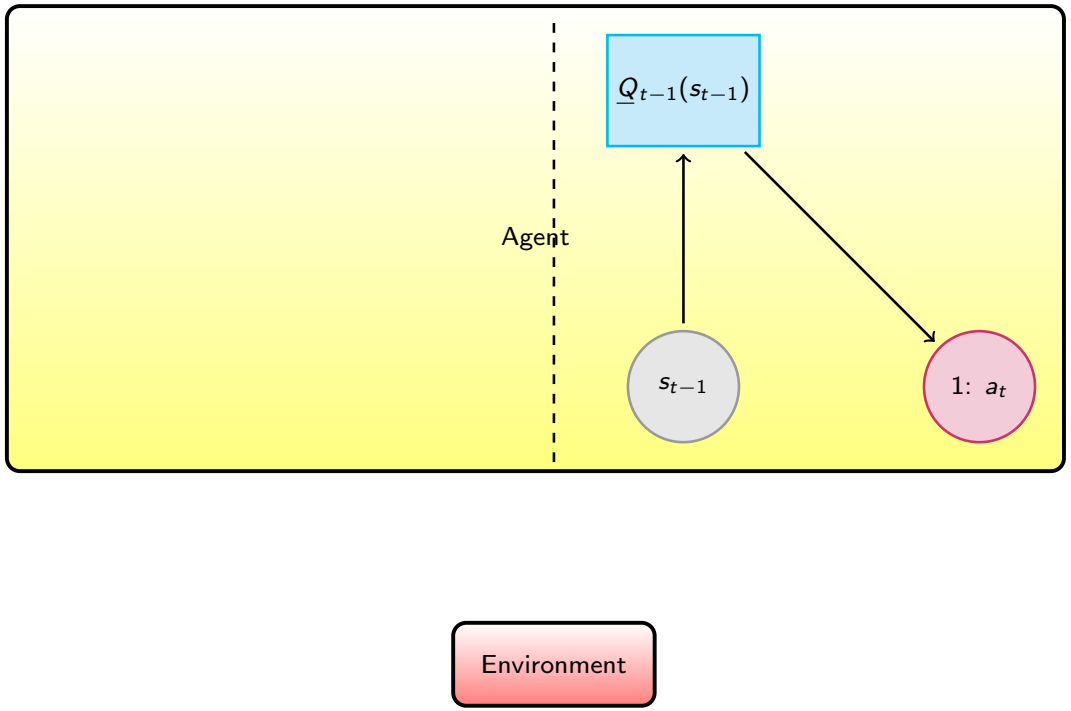
Q-Learning



*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning

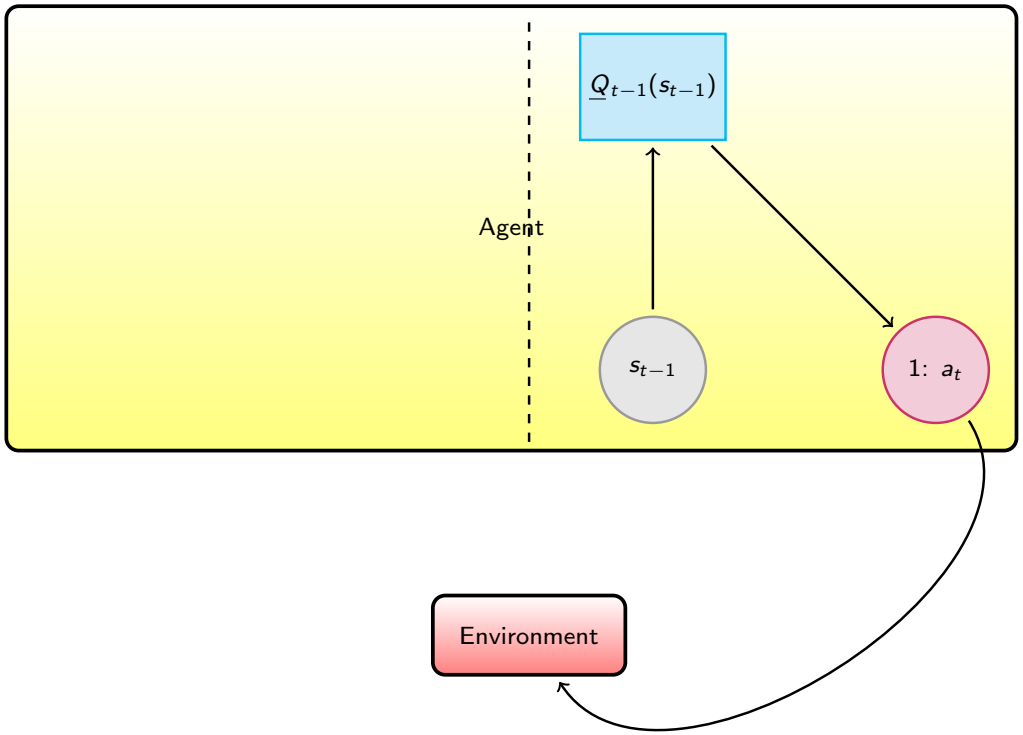


1. Chose **action** by policy: a_t

*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning

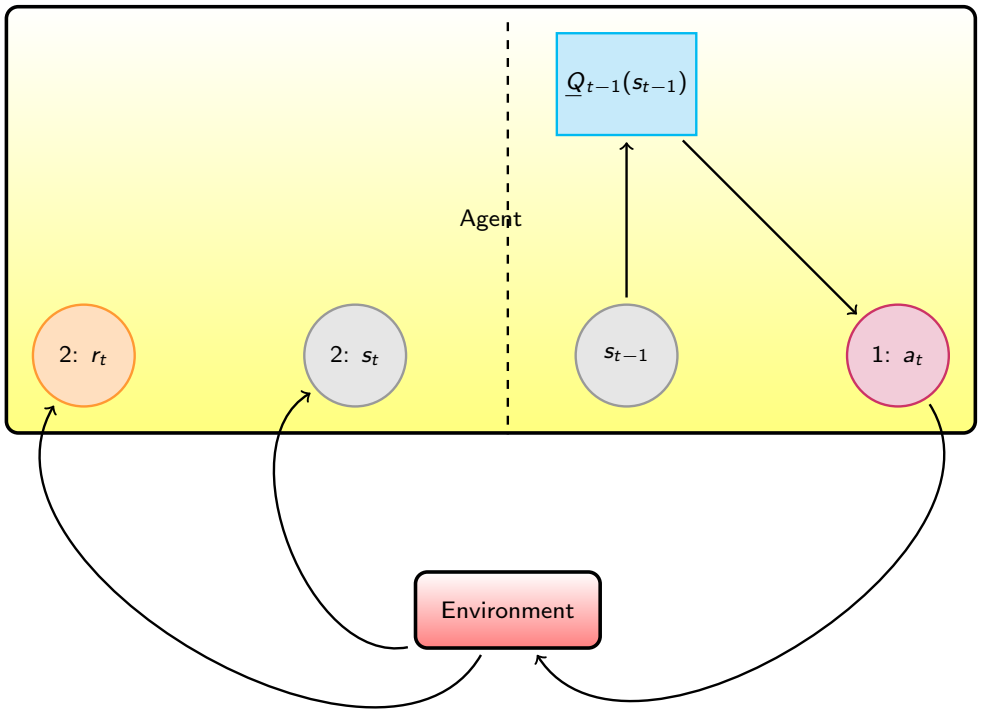


1. Chose **action** by policy: a_t

*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning

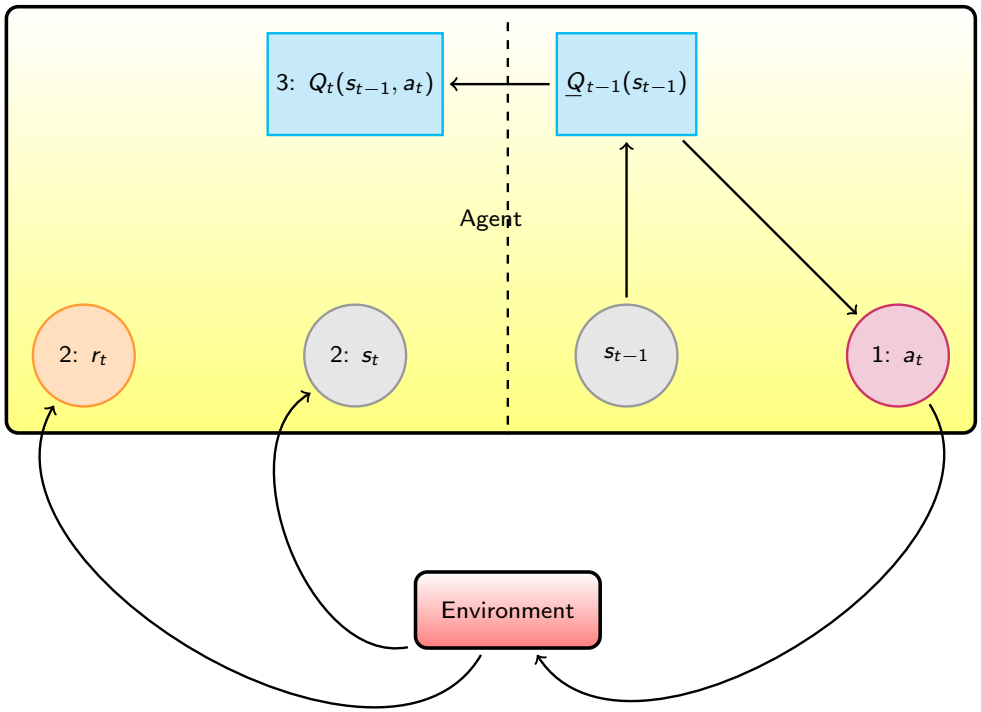


1. Chose **action** by policy: a_t
2. Get **reward** and new **state**: r_t, s_t

*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning



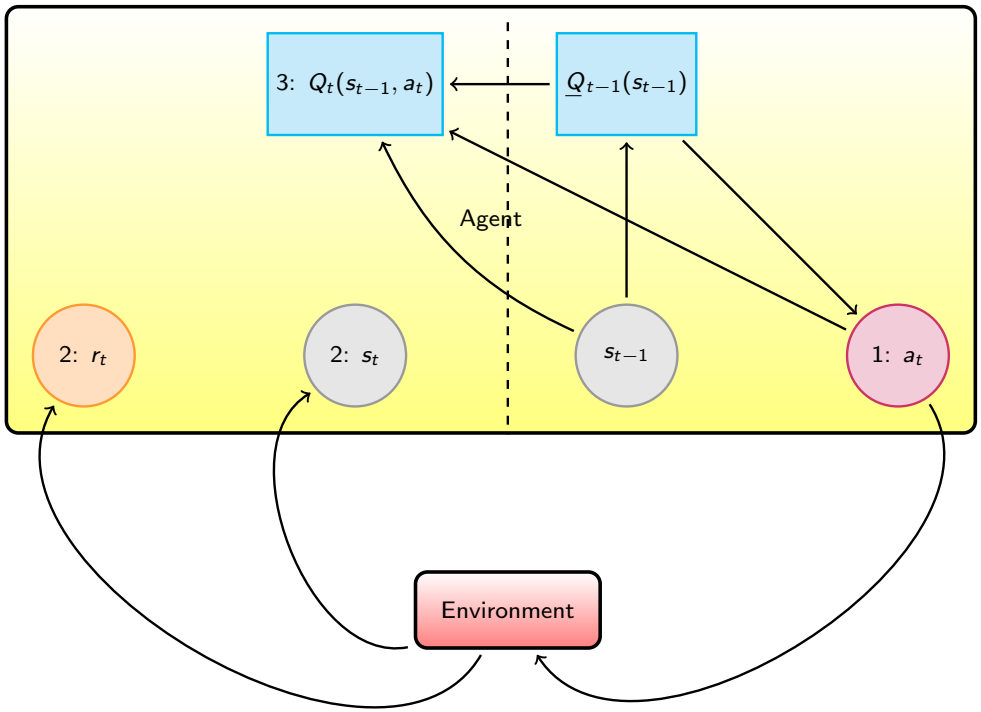
1. Chose **action** by policy: a_t
2. Get **reward** and new **state**: r_t, s_t
3. **Update** the action-value function by the difference of:

$$Q_t \qquad = Q_{t-1} \qquad + \lambda \Big(\qquad - Q_{t-1} \qquad \Big)$$

*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning



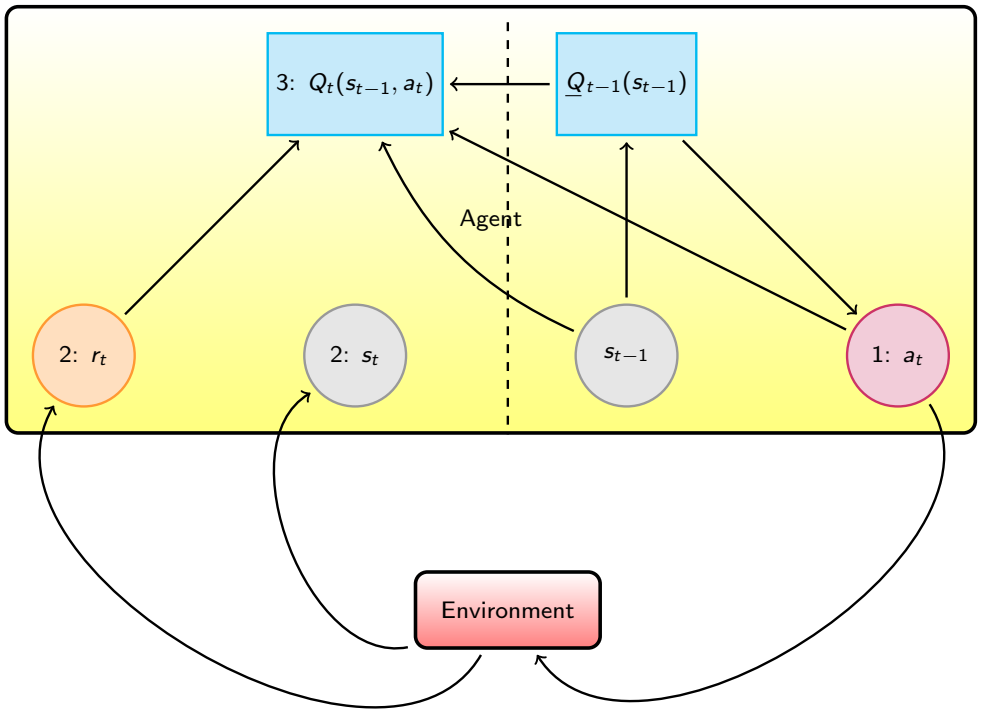
1. Chose **action** by policy: a_t
2. Get **reward** and new **state**: r_t, s_t
3. **Update** the action-value function by the difference of:
   1. The old action-value function

$$Q_t(s_{t-1}, a_t) = Q_{t-1}(s_{t-1}, a_t) + \lambda \Big(\qquad - Q_{t-1}(s_{t-1}, a_t) \Big)$$

*Note change in time boundary. No deeper meaning, we just used it for convenience.

Reinforcement Learning

Q-Learning



1. Chose **action** by policy: a_t
2. Get **reward** and new **state**: r_t, s_t
3. **Update** the action-value function by the difference of:
   1. The old action-value function
   2. The expected return** from the reward

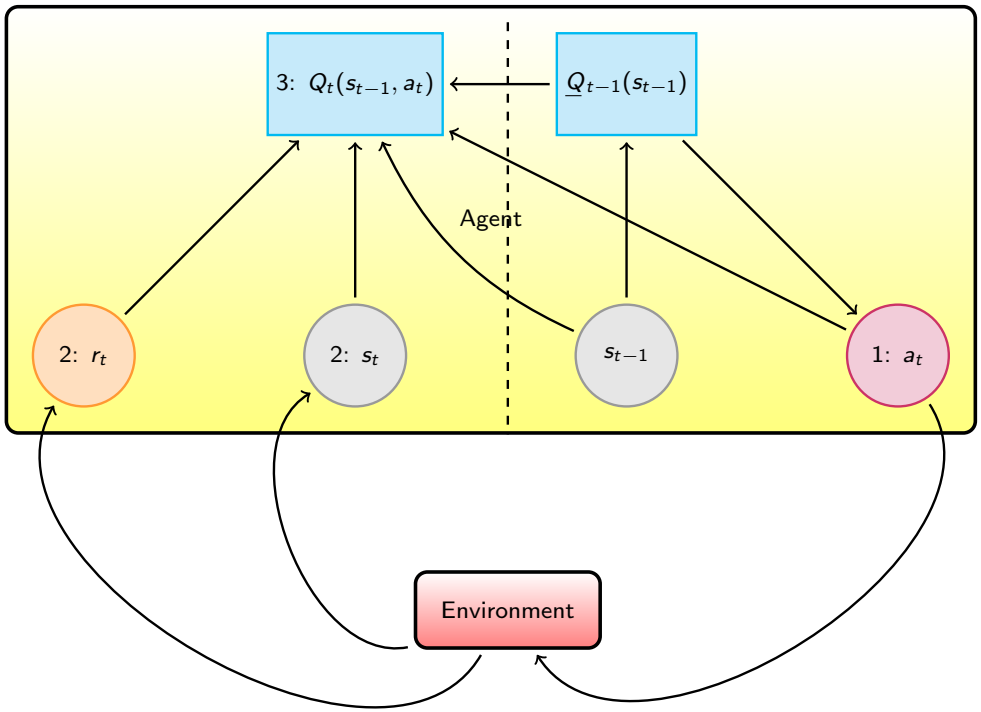
$$Q_t(s_{t-1}, a_t) = Q_{t-1}(s_{t-1}, a_t) + \lambda \left(r_t \right. \qquad \left. - Q_{t-1}(s_{t-1}, a_t) \right)$$

*Note change in time boundary. No deeper meaning, we just used it for convenience.

**Discounted future reward

Reinforcement Learning

Q-Learning



1. Chose **action** by policy: a_t
2. Get **reward** and new **state**: r_t, s_t
3. **Update** the action-value function by the difference of:
   1. The old action-value function
   2. The expected return** from the reward and when following in the future the at moment optimal policy (bootstrapping).

$$Q_t(s_{t-1}, a_t) = Q_{t-1}(s_{t-1}, a_t) + \lambda \left(r_t + \gamma \max_a \underline{Q}_{t-1}(s_t, a) - Q_{t-1}(s_{t-1}, a_t) \right)$$

*Note change in time boundary. No deeper meaning, we just used it for convenience.

**Discounted future reward

Q-Learning the Learning Rate

Overview

In General

- A first reinforcement learner solves the actual task (Agent 1).

Q-Learning the Learning Rate

Overview

In General

- A first reinforcement learner solves the actual task (Agent 1).
- A Q-learning learner learns the optimal learning rate λ_{opt} (Agent 2). Its **action** is the learning rate of Agent 1, its **reward** a feedback from Agent 1.

Q-Learning the Learning Rate

Overview

In General

- A first reinforcement learner solves the actual task (Agent 1).
- A Q-learning learner learns the optimal learning rate λ_{opt} (Agent 2). Its **action** is the learning rate of Agent 1, its **reward** a feedback from Agent 1.

In Our Case (Probabilistic Reversal Learning)

- Agent 1 is a discounted recursive mean learner:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t \underbrace{(y_t - \hat{x}_{t-1})}_{\eta_t},$$

where η_t is the estimation error at time step t.

Q-Learning the Learning Rate

Overview

In General

- A first reinforcement learner solves the actual task (Agent 1).
- A Q-learning learner learns the optimal learning rate λ_{opt} (Agent 2). Its **action** is the learning rate of Agent 1, its **reward** a feedback from Agent 1.

In Our Case (Probabilistic Reversal Learning)

- Agent 1 is a discounted recursive mean learner:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t \underbrace{(y_t - \hat{x}_{t-1})}_{\eta_t},$$

  where η_t is the estimation error at time step t.

- Agent 2 gets as a feedback (reward) η_t^2

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2 \left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1}) \right)$$

  - The set of actions are the possible learning rates for Agent 1.
  - It is a single state learner.
  - The default policy is a Tabu search policy.

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1**

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** → **reward** → **update**.
- **Process 1** and **Process 2**

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.

At time step t

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.

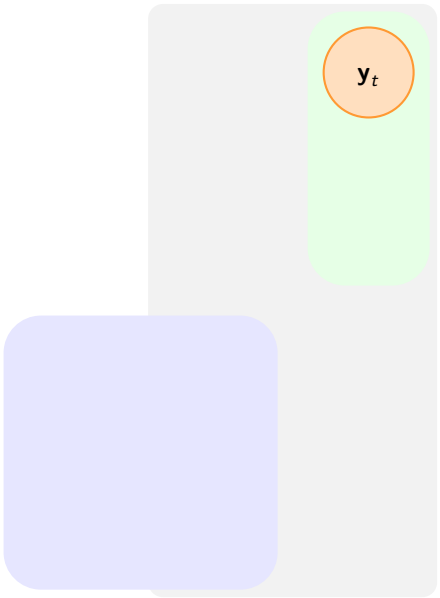
At time step t

a_t -

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

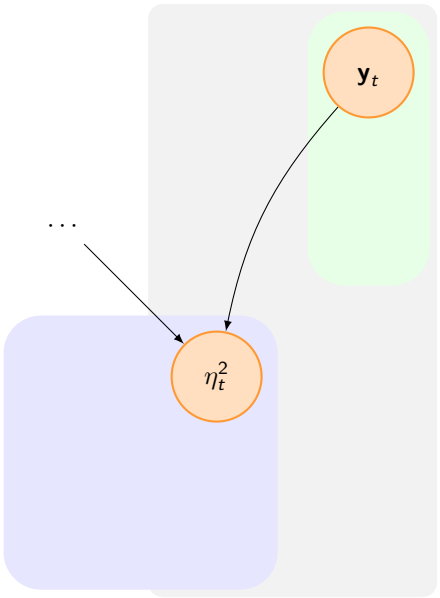
a_t -

r_t It observes y_t

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

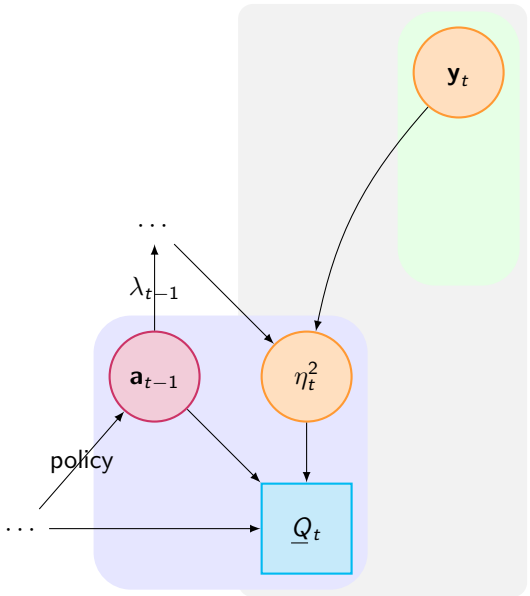
r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

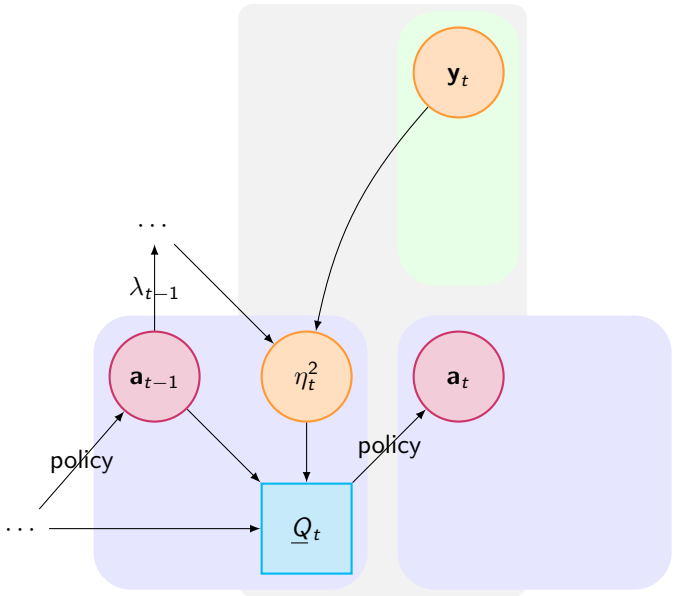
u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

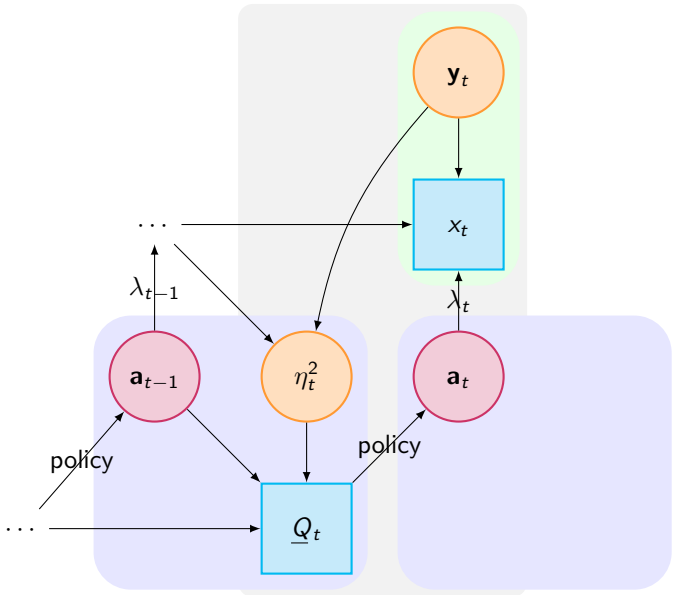
$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

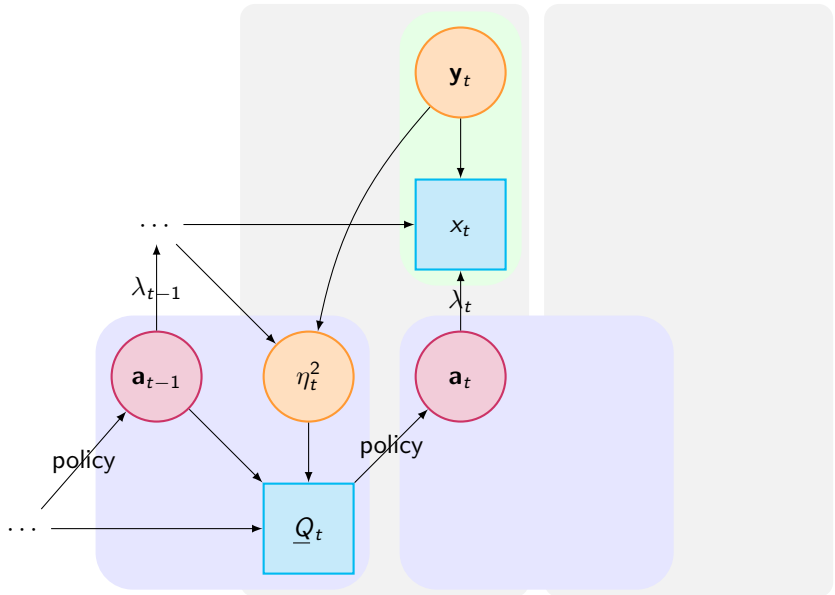
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

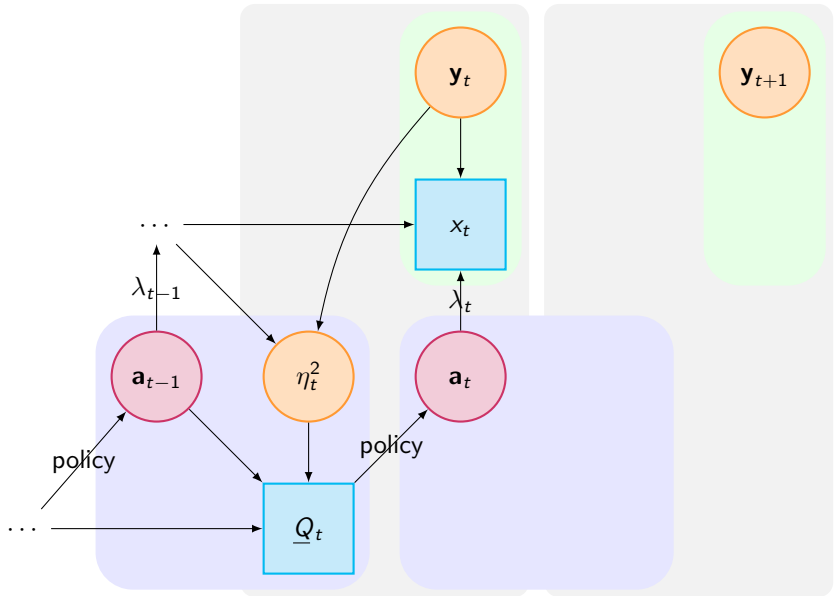
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

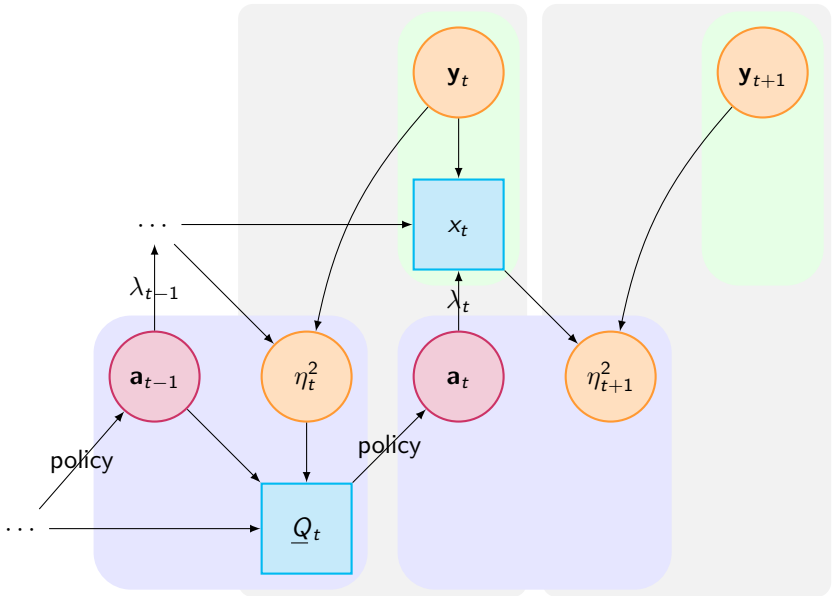
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

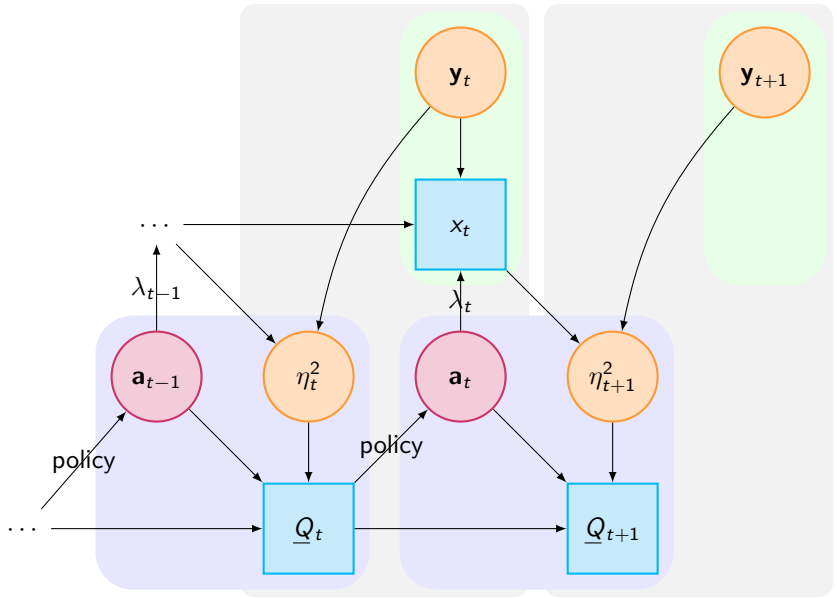
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

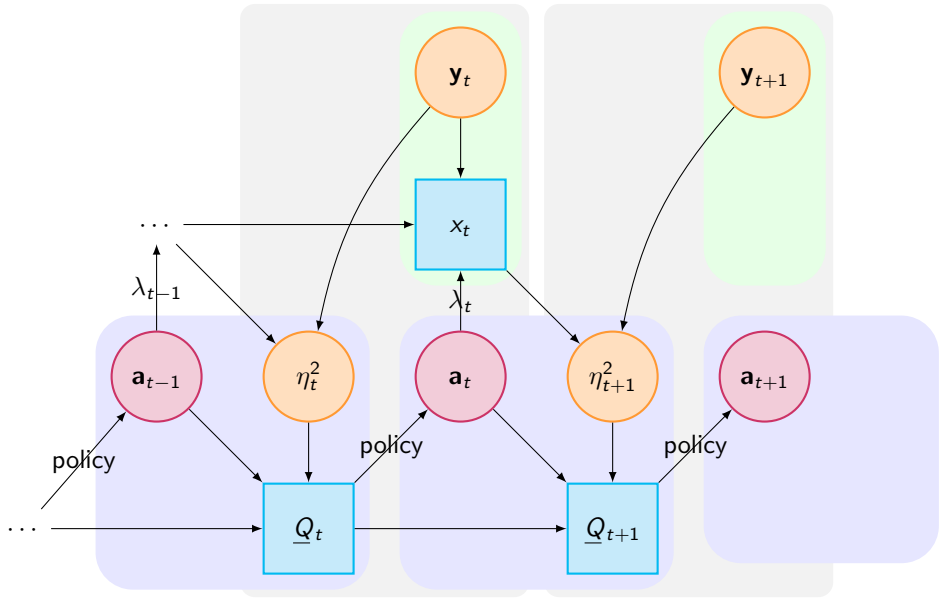
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

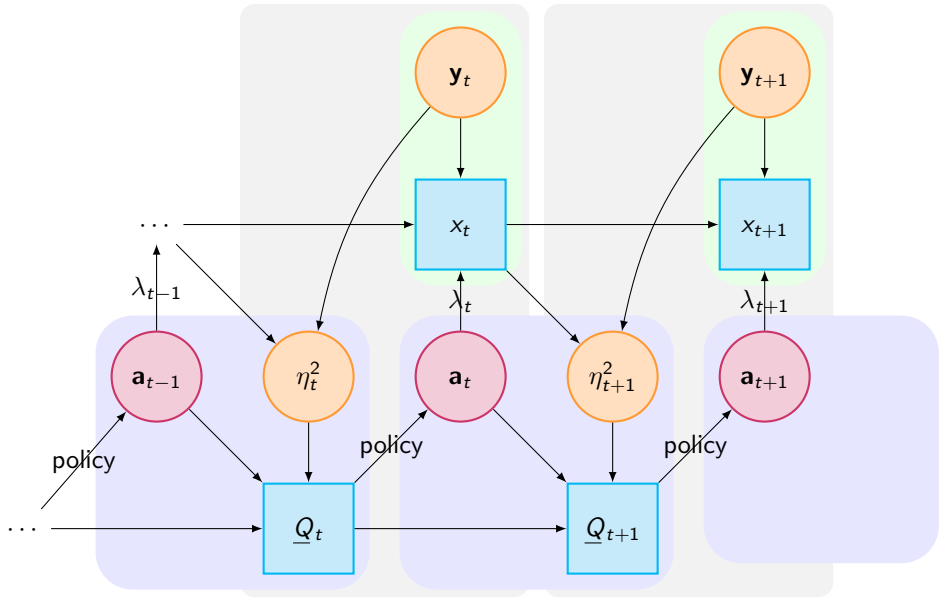
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** $\longrightarrow$ **reward** $\longrightarrow$ **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

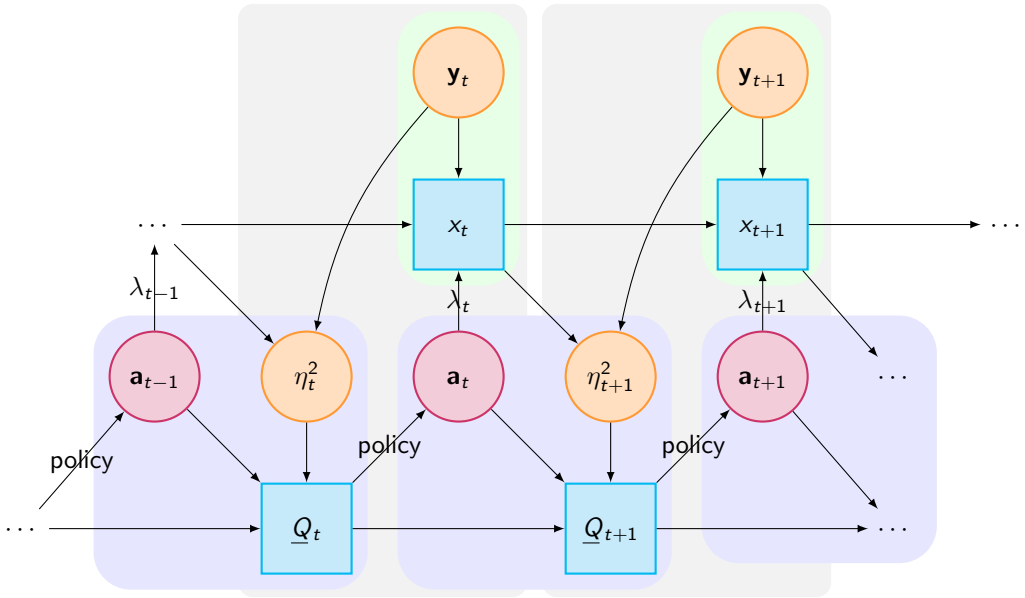
u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Q-Learning the Learning Rate

Implementation

- Form of the RLPs: **action** → **reward** → **update**.
- **Process 1** and **Process 2** interact by r_t (u_t) and $u_t(a_t)$.



At time step t

a_t -

r_t It observes y_t

r_{t-1} It determines the error: $\eta_t = y_t - x_{t-1}$

u_{t-1} It updates its estimate of $\mathbb{E}\left[\eta^2(a_{t-1})\right]$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \lambda 2\left(\eta_t^2 + \gamma 2 \min \underline{Q}_{t-1} - Q_{t-1}(a_{t-1})\right)$$

a_t It determines its new action by policy: $a_t(\underline{Q}_t)$

u_t It updates its estimate of p:

$$\hat{x}_t = \hat{x}_{t-1} + \lambda_t(a_t) \cdot \eta_t$$

Measure of Quality

	Nr.	Error	Available to Learner	Comment
Estimation	1	$E = \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{u}}_t \neq u_t$	no	Reward r maximizer
	2	$E = \frac{1}{T}\sum_{t=1}^{T}(\hat{\mathbf{x}}_t - x_t)^2$	no	
Prediction		$E = \frac{1}{T}\sum_{t=2}^{T} \hat{\mathbf{u}}_t \neq u_{t-1}$	no	
		$E = \frac{1}{T}\sum_{t=2}^{T}(\hat{\mathbf{x}}_t - x_{t-1})^2$	no	
	3	$E = \frac{1}{T}\sum_{t=2}^{T}\underbrace{(y_t - \hat{x}_{t-1})}_{\eta_t}^2$	yes	

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\longrightarrow$ Statistical results.

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

Static Task

One given state (v, p) for one run.

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

Static Task

One given state (v, p) for one run.

Dynamic Task

$C \in \{\quad\}$ times a run a new state (p, v) is set by random.

Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\longrightarrow$ Statistical results.

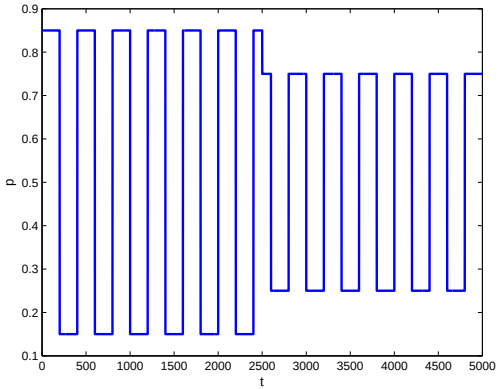
Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

Static Task

One given state (v, p) for one run.

Dynamic Task

$C \in \{2 \quad\}$ times a run a new state (p, v) is set by random.



Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

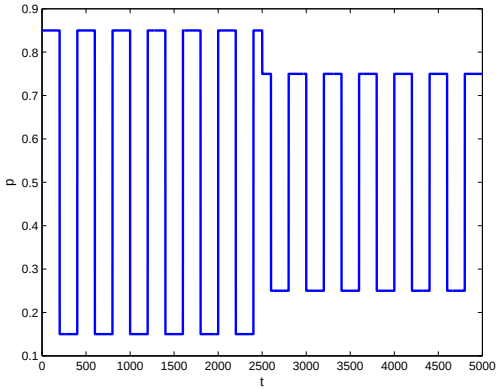
Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

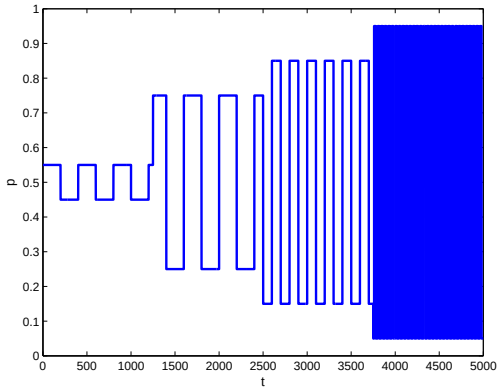
Static Task

One given state (v, p) for one run.

Dynamic Task

$C \in \{2, 4\}$ times a run a new state (p, v) is set by random.





Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

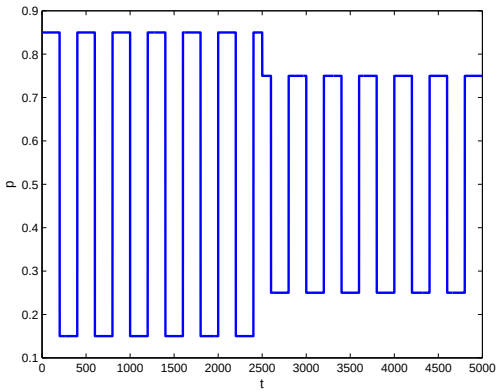
Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

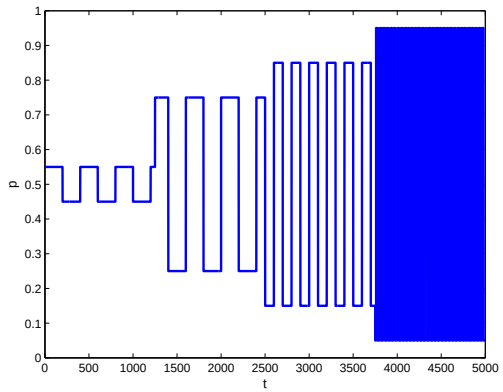
Static Task

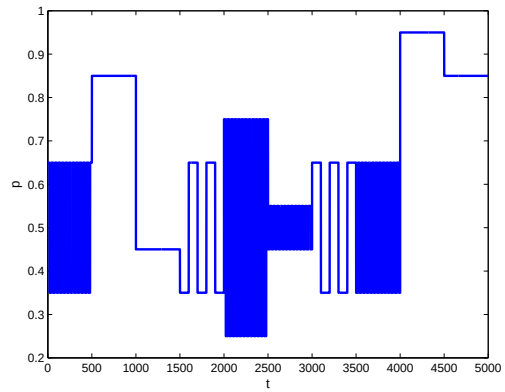
One given state (v, p) for one run.

Dynamic Task

$C \in \{2, 4, 10\}$ times a run a new state (p, v) is set by random.







Task

Volatility Frequency of change of the better stimuli: $v \in \{0.001, 0.005, 0.01, 0.05\}$

Risk Reliability of observation $p = \Pr\{\mathbf{y}_t = u_t\}$: $p \in \{0.55, 0.65, 0.75, 0.85, 0.95\}$

Run Consist of $T = 5000$ time-steps. Repeated $\rightarrow$ Statistical results.

Learning Rates $\lambda \in \{0.01, 0.11, 0.21, 0.31, 0.41, 0.51, 0.61, 0.71, 0.81, 0.91\}$

Static Task

One given state (v, p) for one run.

Dynamic Task

$C \in \{2, 4, 10\}$ times a run a new state (p, v) is set by random.

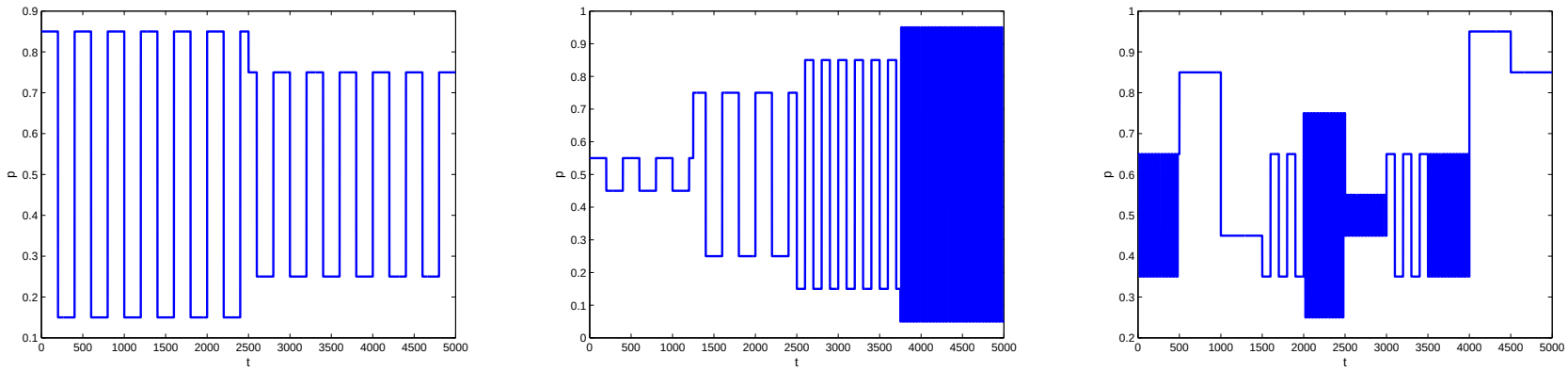
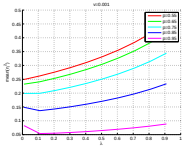


Figure: Example stimuli dynamic task

Fixed Learning Rates

Running learner 1 for fixed learning rates on the static task



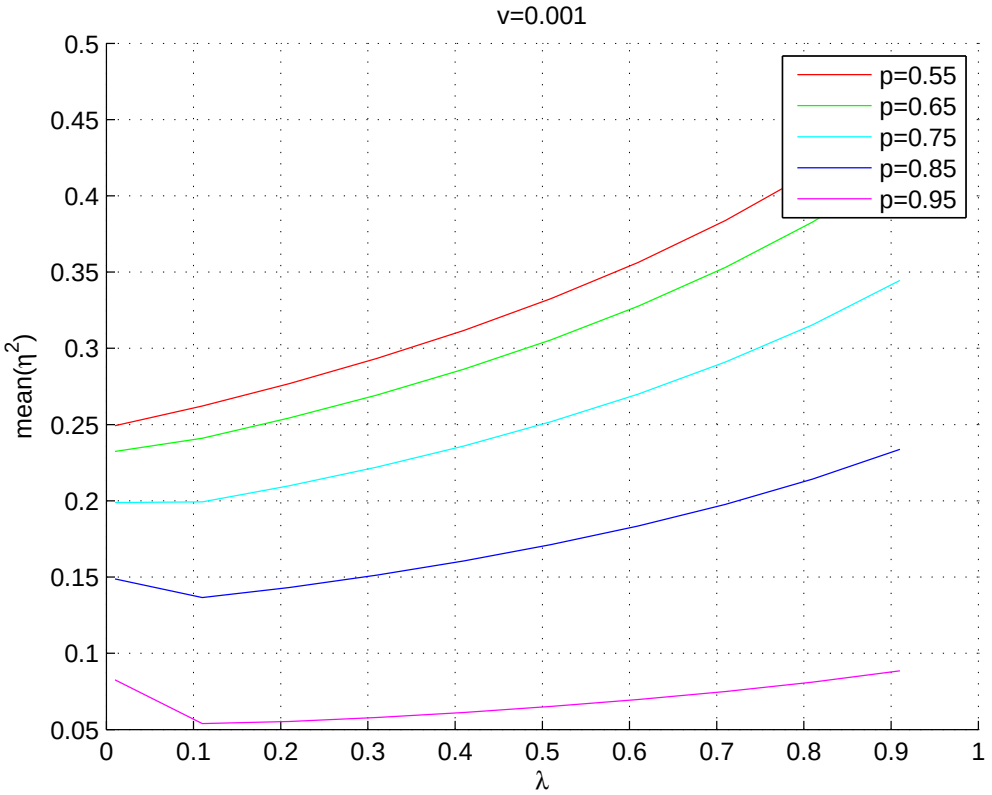
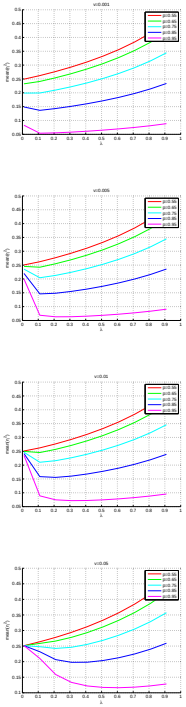


Figure: Error 3 for fixed learning rates

Fixed Learning Rates

Running learner 1 for fixed learning rates on the static task



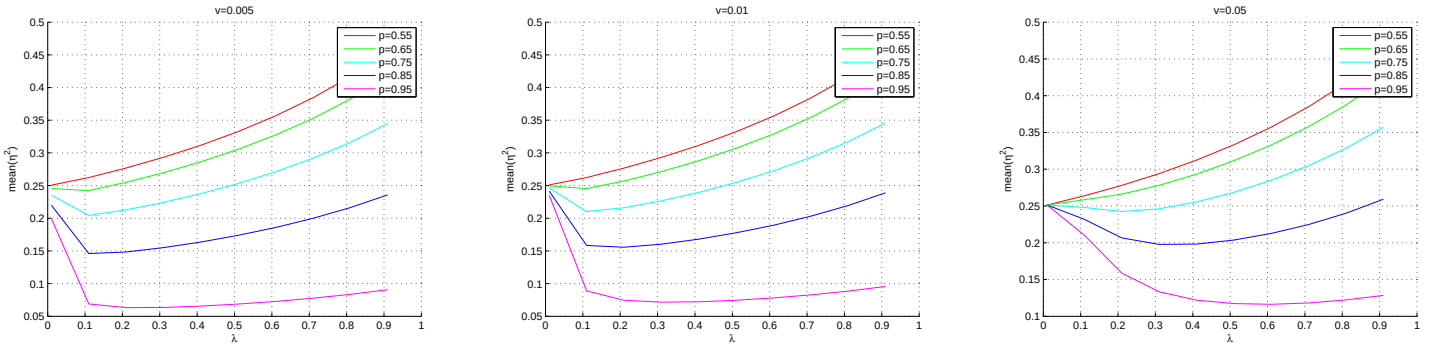
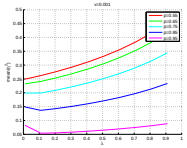
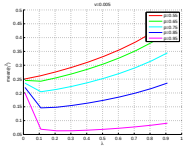


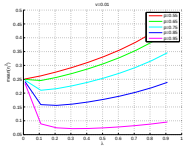
Figure: ... similar for other volatilities

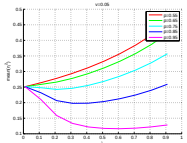
Fixed Learning Rates

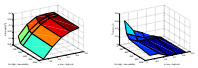
Running learner 1 for fixed learning rates on the static task

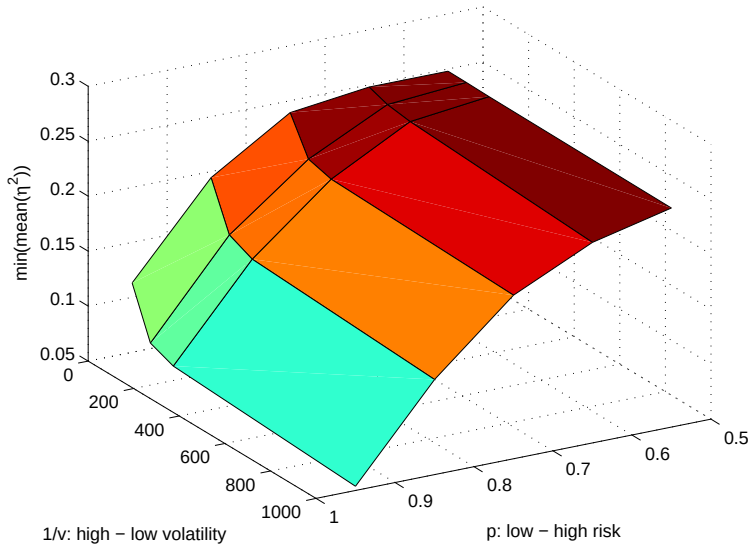




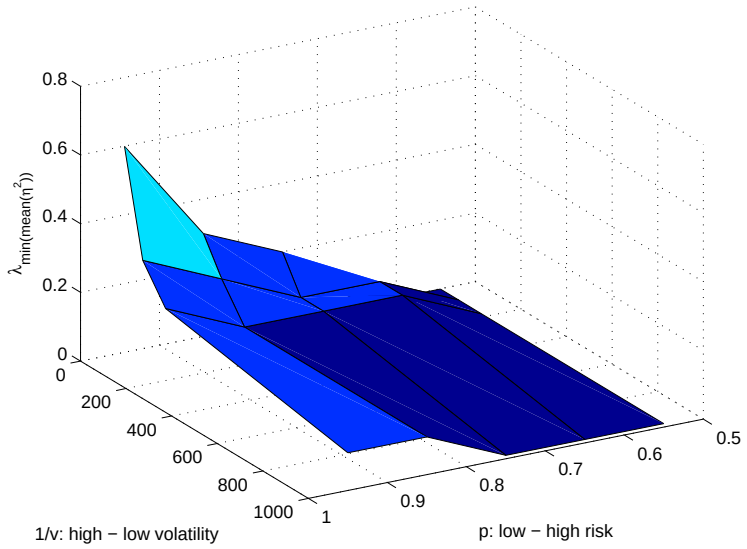








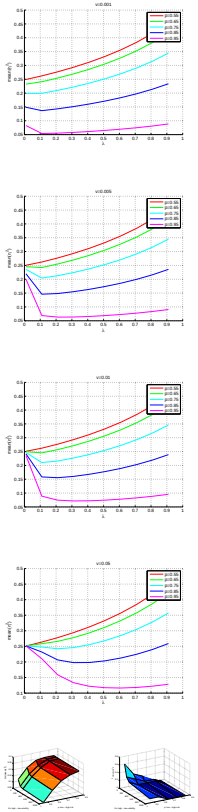
$E_3(\lambda_{opt})$

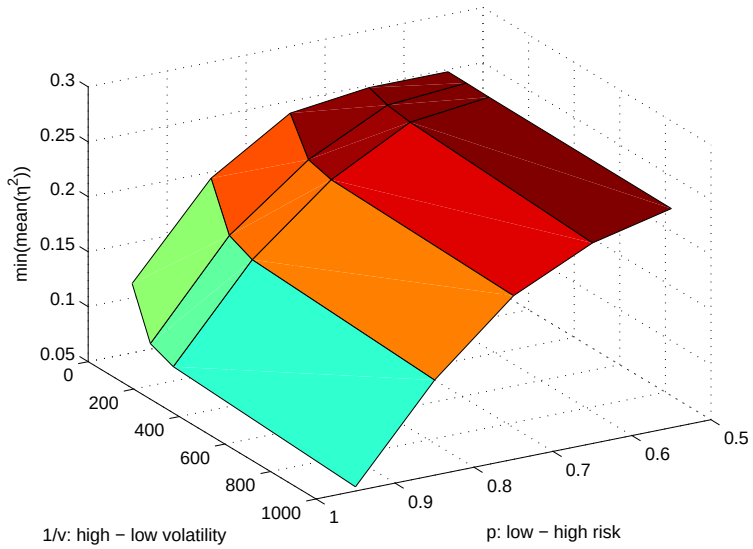


λ_{opt}

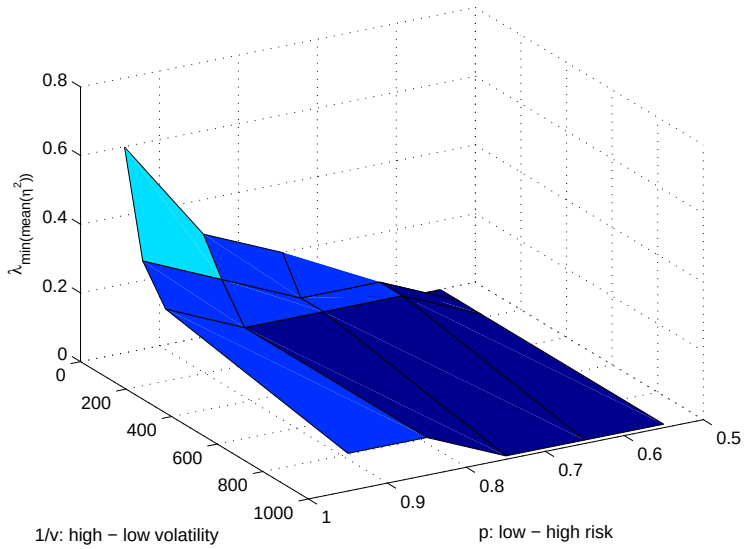
Fixed Learning Rates

Running learner 1 for fixed learning rates on the static task





$E_3(\lambda_{opt})$

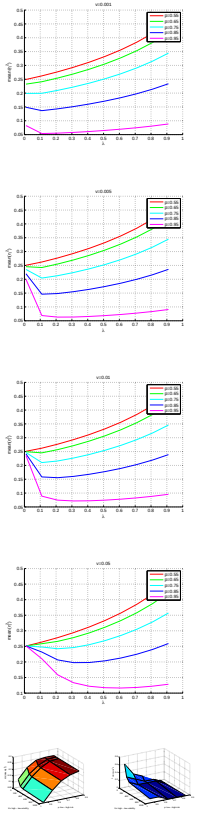


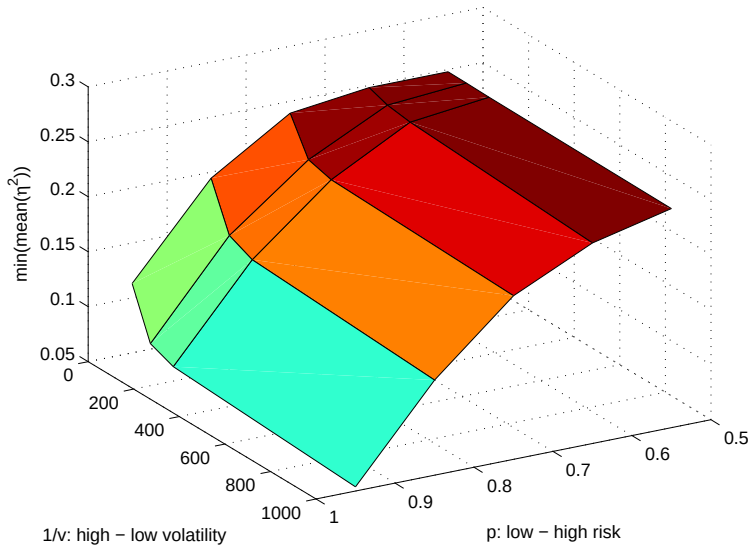
λ_{opt}

- Behavior according risk and volatility is given by $\lambda_{opt}(v, p)$

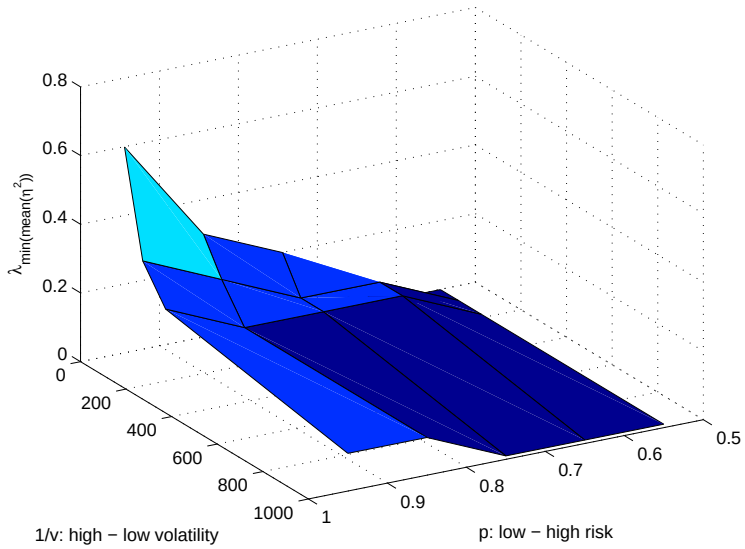
Fixed Learning Rates

Running learner 1 for fixed learning rates on the static task





$E_3(\lambda_{opt})$



λ_{opt}

- Behavior according risk and volatility is given by $\lambda_{opt}(v, p)$
- mean(η^2) are convex functions
- Get an idea of Q_{init}
- $\lambda_{opt}(v, p)$ is concavel

Testing

Quite a lot of things to test:

Testing

Quite a lot of things to test:

Initial Values

- Initial Q-Values (Agent 2)

Policy (Agent 2)

- Tabu-Search Policy (permutation, ban)
- Neighborhood Policy (width and shape of neighborhood)
- ϵ-Greedy Policy (ϵ)
- Softmin Policy (τ)

Update (Agent 2)

- Discount rate $\gamma 2 \rightarrow (= 0$: discounted mean$(\eta^2))$
- Learning rate $\lambda 2$
- Neighborhood Q-Update

Others

- Possible learning rates (Agent 1)
- Other methods than two agent setup

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)

Results

Primary

Initial Values $\underline{Q}_{init}$

! Need to be above a bound (we do minimization)

→ We gave estimates for bound

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Two crucial facts for a good learner:

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Two crucial facts for a good learner:

Markov Property of 2nd Learner

- Small learning rate (eg. $\lambda_t = 0.01$) $\longrightarrow$ η_t^2 is sparely dependent on λ_t)

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Two crucial facts for a good learner:

Markov Property of 2nd Learner

- Small learning rate (eg. $\lambda_t = 0.01$) $\longrightarrow$ η_t^2 is sparely dependent on λ_t)

Convexity of $\mathbb{E}\left[\eta^2\right]$

- rc By fixed learning rates experiments:

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Two crucial facts for a good learner:

Markov Property of 2nd Learner

- Small learning rate (eg. $\lambda_t = 0.01$) $\longrightarrow$ η_t^2 is sparely dependent on λ_t)

Convexity of $\mathbb{E}\left[\eta^2\right]$

- rc By fixed learning rates experiments:
- → One can search within the neighborhood of a learning rate .

Results

Primary

Initial Values $\underline{Q}_{init}$

- ! Need to be above a bound (we do minimization)
- → We gave estimates for bound
- → In original paper their were not: opposite learning rate behavior according to risk and volatility

Two crucial facts for a good learner:

Markov Property of 2nd Learner

- Small learning rate (eg. $\lambda_t = 0.01$) $\longrightarrow$ η_t^2 is sparely dependent on λ_t)

Convexity of $\mathbb{E}\left[\eta^2\right]$

- rc By fixed learning rates experiments:
- → One can search within the neighborhood of a learning rate .
- → Helps to overcome the Markov issue.

Results

Secondary

Policy

- Neighborhood: tabu-search and neighborhood search $\gg$ ϵ-greedy and softmin
- Tabu Policy: Results show no benefit from banning

Results

Secondary

Policy

- Neighborhood: tabu-search and neighborhood search $\gg$ ϵ-greedy and softmin
- Tabu Policy: Results show no benefit from banning

Update

- Results show no benefit from Q-learning over simple discounted recursive mean learning.
- $\lambda 2$ depend on the dynamics of the task.

Results

Secondary

Policy

- Neighborhood: tabu-search and neighborhood search $\gg$ ϵ-greedy and softmin
- Tabu Policy: Results show no benefit from banning

Update

- Results show no benefit from Q-learning over simple discounted recursive mean learning.
- $\lambda 2$ depend on the dynamics of the task.

Others

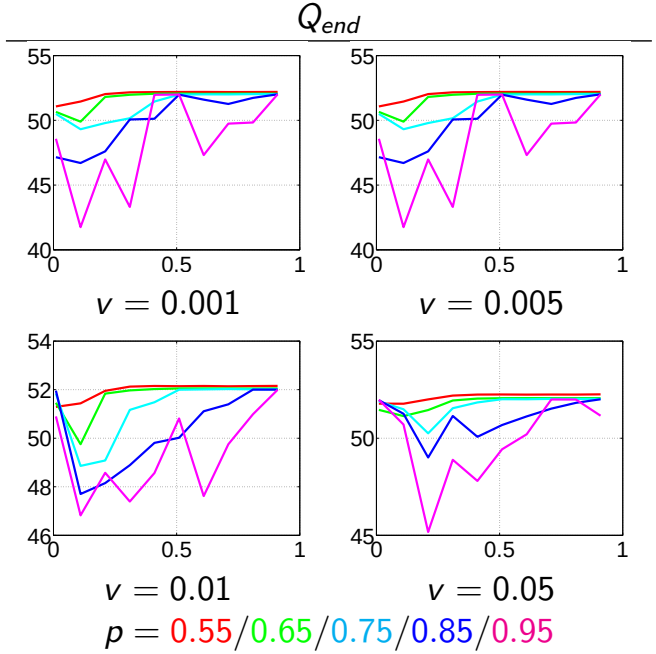
- Better performance with different learning rate set.
- Concurrent learner superior

Results

General Observations on the Task

Tabu Policy

Learner			Results				
	v \ p		0.55	0.65	0.75	0.85	0.95
Q-Learning	λ_{avg}	0.001	0.108	0.117	0.219	0.237	0.313
$Q_{init} = 52$		0.005	0.113	0.104	0.225	0.310	0.290
$\lambda 2 = 0.1$		0.01	0.111	0.117	0.212	0.330	0.296
$\gamma 2 = 0.995$		0.05	0.141	0.143	0.236	0.373	0.323
	η^2_{avg}	0.001	0.269	0.244	0.214	0.154	0.062
Tabu-Policy		0.005	0.271	0.249	0.216	0.169	0.085
boundary $= 8$		0.01	0.270	0.250	0.220	0.168	0.116
$\sigma^2 = 0.25^2$		0.05	0.278	0.266	0.246	0.210	0.148
	$\lvert x - \hat{x} \rvert_{avg}$	0.001	0.111	0.097	0.124	0.107	0.066
		0.005	0.119	0.116	0.130	0.152	0.108
		0.01	0.119	0.117	0.138	0.138	0.161
		0.05	0.137	0.152	0.172	0.183	0.161
	right est.	0.001	68%	88%	92%	95%	98%
		0.005	62%	80%	91%	90%	94%
		0.01	60%	79%	89%	92%	88%
		0.05	54%	64%	79%	85%	90%

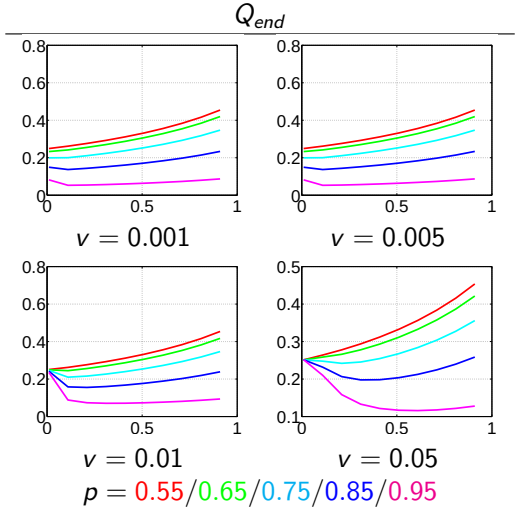


Results

General Observations on the Task

Tabu Policy | Concurrent Learner

Learner	Results						
Concurrent Learning $\lambda 2 = \frac{1}{t}$ Policy $\underset{a}{\text{argmin}}\ \eta^2$	v \ p		0.55	0.65	0.75	0.85	0.95
	λ_{avg}	0.001	0.011	0.013	0.050	0.107	0.118
		0.005	0.011	0.097	0.110	0.113	0.247
		0.01	0.011	0.102	0.111	0.208	0.329
		0.05	0.011	0.012	0.211	0.346	0.603
	η^2_{avg}	0.001	0.249	0.233	0.199	0.137	0.053
		0.005	0.250	0.242	0.205	0.146	0.062
		0.01	0.251	0.244	0.210	0.155	0.071
		0.05	0.251	0.251	0.241	0.197	0.116
	$\lvert x - \hat{x} \rvert_{avg}$	0.001	0.033	0.048	0.068	0.071	0.048
		0.005	0.047	0.101	0.095	0.089	0.070
		0.01	0.050	0.106	0.106	0.115	0.079
		0.05	0.052	0.149	0.161	0.162	0.095
	right est.*	0.001	85%	94%	95%	99%	100%
		0.005	62%	85%	96%	97%	99%
		0.01	56%	82%	93%	96%	98%
		0.05	51%	52%	80%	89%	95%

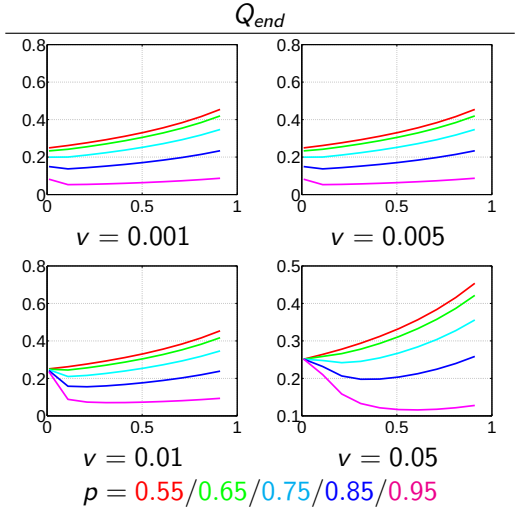


Results

General Observations on the Task

Tabu Policy | Concurrent Learner | Comparison

Learner			Results				
Concurrent Learning $\lambda 2 = \frac{1}{t}$ Policy $\underset{a}{\text{argmin}}\ \eta^2$	v \ p		0.55	0.65	0.75	0.85	0.95
	λ_{avg}	0.001	0.011	0.013	0.050	0.107	0.118
		0.005	0.011	0.097	0.110	0.113	0.247
		0.01	0.011	0.102	0.111	0.208	0.329
		0.05	0.011	0.012	0.211	0.346	0.603
	η^2_{avg}	0.001	0.249 (93%)	0.233 (95%)	0.199 (93%)	0.137 (89%)	0.053 (85%)
		0.005	0.250 (92%)	0.242 (97%)	0.205 (95%)	0.146 (86%)	0.062 (73%)
		0.01	0.251 (93%)	0.244 (98%)	0.210 (95%)	0.155 (92%)	0.071 (61%)
		0.05	0.251 (90%)	0.251 (94%)	0.241 (98%)	0.197 (94%)	0.116 (78%)
	$\lvert x - \hat{x} \rvert_{avg}$	0.001	0.033 (30%)	0.048 (49%)	0.068 (55%)	0.071 (66%)	0.048 (72%)
		0.005	0.047 (39%)	0.101 (87%)	0.095 (73%)	0.089 (58%)	0.070 (64%)
		0.01	0.050 (42%)	0.106 (90%)	0.106 (77%)	0.115 (83%)	0.079 (49%)
		0.05	0.052 (38%)	0.149 (98%)	0.161 (93%)	0.162 (89%)	0.095 (59%)
	right est.*	0.001	85% (428%)	94% (143%)	95% (121%)	99% (130%)	100% (113%)
		0.005	62% (92%)	85% (144%)	96% (128%)	97% (149%)	99% (134%)
		0.01	56% (39%)	82% (122%)	93% (125%)	96% (128%)	98% (184%)
		0.05	51% (2%)	52% (2%)	80% (112%)	89% (130%)	95% (137%)



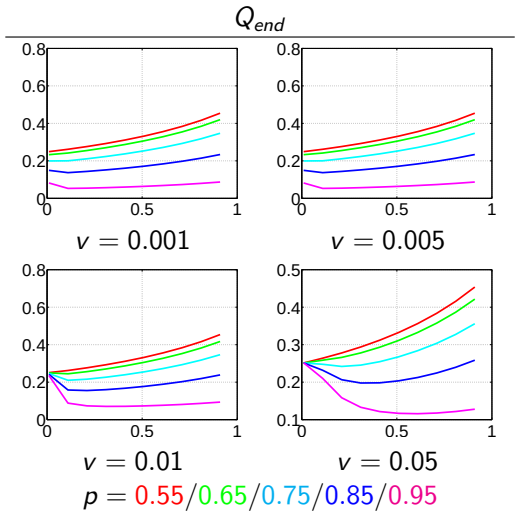
* (percentages) calculated by preprocessing

Results

General Observations on the Task

Tabu Policy | Concurrent Learner | Comparison | Consequences

Learner		v \ p	0.55	0.65	0.75	0.85	0.95
Concurrent Learning	λ_{avg}	0.001	0.011	0.013	0.050	0.107	0.118
$\lambda 2 = \frac{1}{t}$		0.005	0.011	0.097	0.110	0.113	0.247
Policy		0.01	0.011	0.102	0.111	0.208	0.329
$\underset{a}{\text{argmin}}\ \eta^2$		0.05	0.011	0.012	0.211	0.346	0.603
	η^2_{avg}	0.001	0.249 (93%)	0.233 (95%)	0.199 (93%)	0.137 (89%)	0.053 (85%)
		0.005	0.250 (92%)	0.242 (97%)	0.205 (95%)	0.146 (86%)	0.062 (73%)
		0.01	0.251 (93%)	0.244 (98%)	0.210 (95%)	0.155 (92%)	0.071 (61%)
		0.05	0.251 (90%)	0.251 (94%)	0.241 (98%)	0.197 (94%)	0.116 (78%)
	$\lvert x - \hat{x} \rvert_{avg}$	0.001	0.033 (30%)	0.048 (49%)	0.068 (55%)	0.071 (66%)	0.048 (72%)
		0.005	0.047 (39%)	0.101 (87%)	0.095 (73%)	0.089 (58%)	0.070 (64%)
		0.01	0.050 (42%)	0.106 (90%)	0.106 (77%)	0.115 (83%)	0.079 (49%)
		0.05	0.052 (38%)	0.149 (98%)	0.161 (93%)	0.162 (89%)	0.095 (59%)
	right est.*	0.001	85% (428%)	94% (143%)	95% (121%)	99% (130%)	100% (113%)
		0.005	62% (92%)	85% (144%)	96% (128%)	97% (149%)	99% (134%)
		0.01	56% (39%)	82% (122%)	93% (125%)	96% (128%)	98% (184%)
		0.05	51% (2%)	52% (2%)	80% (112%)	89% (130%)	95% (137%)



Minimizing $\eta^2 \not\Rightarrow$ Better estimation of the token = Higher Reward

* (percentages) calculated by preprocessing

Results

General Observations on the Task

Tabu Policy | Concurrent Learner | Comparison | Consequences

Different λ_{opt}

- Minimizing $\eta^2 \not\Rightarrow$ Better estimation of the better stimuli (higher reward)

Results

General Observations on the Task

Different λ_{opt}

- Minimizing $\eta^2 \not\Rightarrow$ Better estimation of the better stimuli (higher reward)
- Experiments with fixed learning rates show: $\lambda_{optError1} \neq \lambda_{optError3}$

Results

General Observations on the Task

Different λ_{opt}

- Minimizing $\eta^2 \not\Rightarrow$ Better estimation of the better stimuli (higher reward)
- Experiments with fixed learning rates show: $\lambda_{optError1} \neq \lambda_{optError3}$

Concurrent Learner

- Concurrent learner superior

Results

General Observations on the Task

Different λ_{opt}

- Minimizing $\eta^2 \not\Rightarrow$ Better estimation of the better stimuli (higher reward)
- Experiments with fixed learning rates show: $\lambda_{optError1} \neq \lambda_{optError3}$

Concurrent Learner

- Concurrent learner superior

$\Rightarrow$

Experiment

This and also the fact that a simple concurrent learner performs significant better demands for thinking about

- The purpose of the experiment
- The experiment itself

Motivation

How do human learners treat risk and volatility?

Does reinforcement learning benefit from Q-learning the learning rate?

Motivation

How do human learners treat risk and volatility?

~~Does reinforcement learning benefit from Q-learning the learning rate?~~

Motivation

How do human learners treat risk and volatility?

$\exists \lambda_{opt}$: Assuming this includes reinforcement learning, e.g. the existence of a learning rate:

~~Does reinforcement learning benefit from Q-learning the learning rate?~~

Motivation

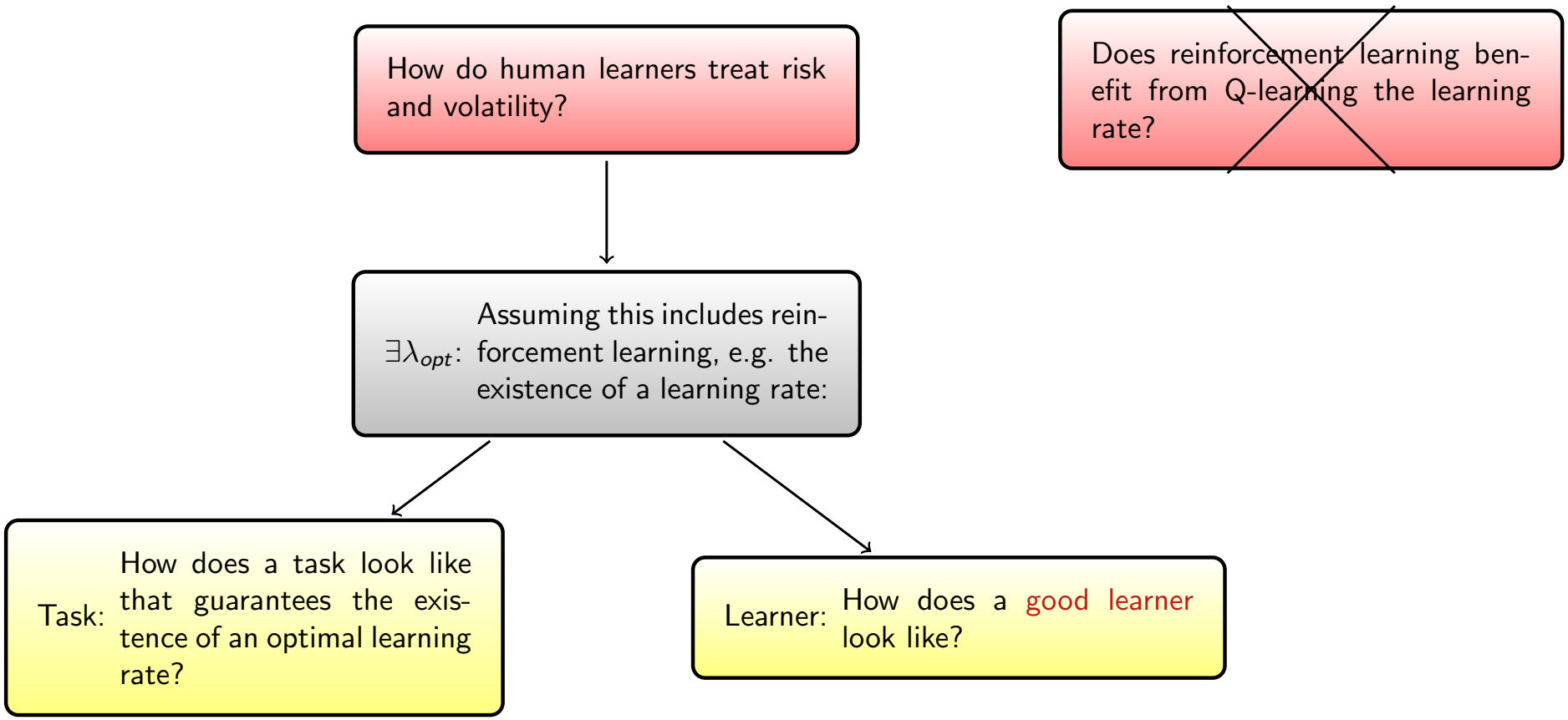
How do human learners treat risk and volatility?

$\exists \lambda_{opt}$: Assuming this includes reinforcement learning, e.g. the existence of a learning rate:

Task: How does a task look like that guarantees the existence of an optimal learning rate?

~~Does reinforcement learning benefit from Q-learning the learning rate?~~

Motivation


How do human learners treat risk and volatility?
Does reinforcement learning benefit from Q-learning the learning rate?
$\exists\lambda_{opt}$: Assuming this includes reinforcement learning, e.g. the existence of a learning rate:
Task: How does a task look like that guarantees the existence of an optimal learning rate?
Learner: How does a good learner look like?

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

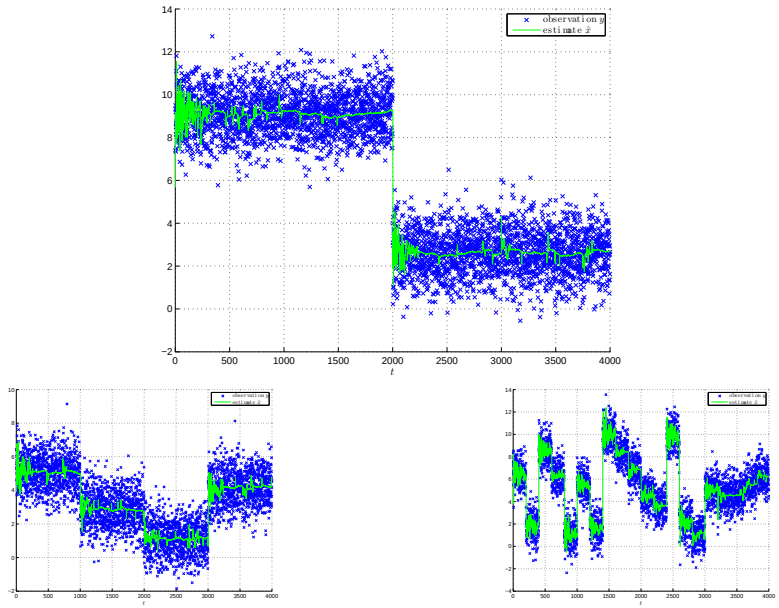
- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.



Realization

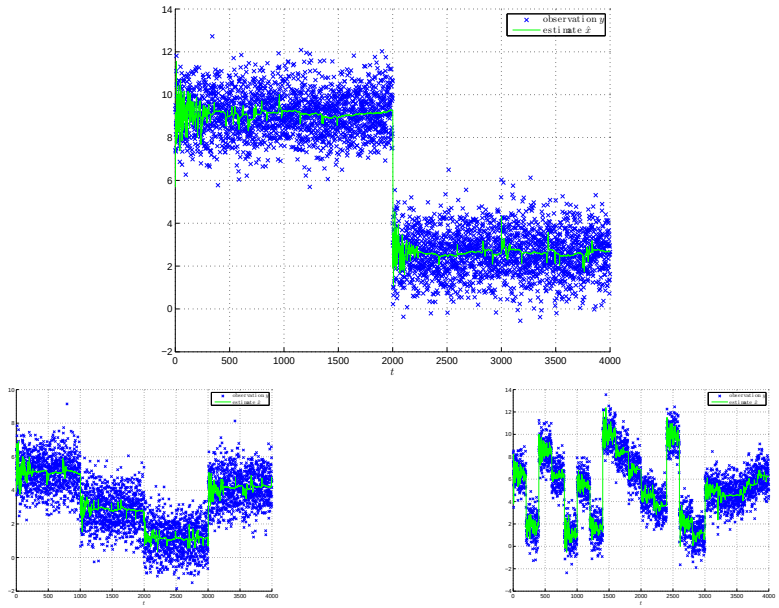
- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

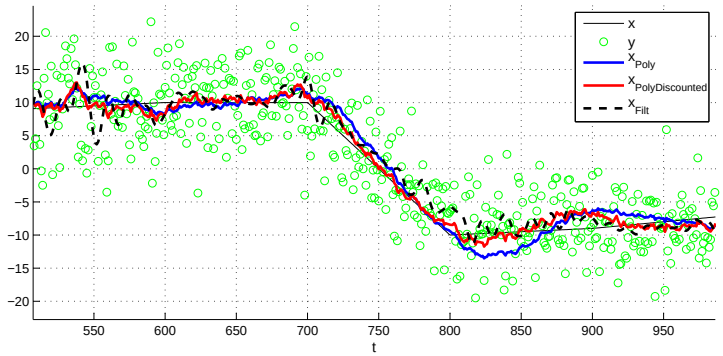
Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.



Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes



Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance

? or rather the estimate?

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance

? or rather the estimate?

? What is a good 'recent'?

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance
- ? or rather the estimate?
- ? What is a good 'recent'?

3-Layer learner:

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance
- ? or rather the estimate?
- ? What is a good 'recent'?

3-Layer learner:

- Learning the whole procedure for different 'recents' ($\lambda 2$)

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T} \sum_{t=1}^{T} (x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance
- ? or rather the estimate?
- ? What is a good 'recent'?

3-Layer learner:

- Learning the whole procedure for different 'recents' ($\lambda 2$)
- Chose on their performance...

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance
- ? or rather the estimate?
- ? What is a good 'recent'?

3-Layer learner:

- Learning the whole procedure for different 'recents' ($\lambda 2$)
- Chose on their performance...

It is not just stacking learners on top of each other

- On has to be careful in how to set the estimate $\hat{x}_t$.
- Its rather a strategy learning...

Task, Realization & Learning

Example Task

Given	hidden truth x
Observe	noisy $\mathbf{y}$
Learn	$\hat{\mathbf{x}}$ by doing estimation in a recursive way by a learning rate
Risk	amount of noise $\mathbf{w}$
Volatility	the change of x

Design the environment in a way such that $\lambda_{\text{opt}}(r, v)$ as a function of risk and volatility changes in time and hence has to be learned.

Realization

- Draw x $C = \{2, 4, 20\}$ times a run from a uniform distribution: $x \sim U(0, 10)$
- Add white Gaussian noise: $\mathbf{y}_t = x_t + \underbrace{\mathbf{w}_t}_{\sim N(0,1)}$
- MoQ: $E = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2 = \frac{||\underline{x} - \underline{\hat{x}}||_2^2}{T}$

Further: Slopes

Learning

Concurrent Learner:

- Learn with all learning rates together ($\rightarrow \#\lambda$ estimates)
- Learn mean(η^2) for all learning rates
  - ? discounted? recursive mean learning
- Chose learning rate with the recent best performance
- ? or rather the estimate?
- ? What is a good 'recent'?

3-Layer learner:

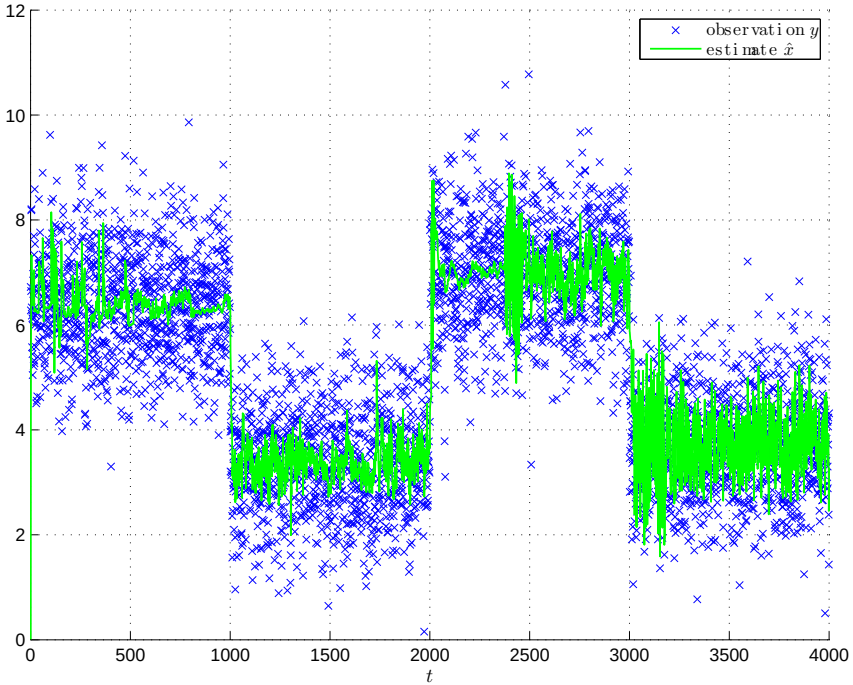
- Learning the whole procedure for different 'recents' ($\lambda 2$)
- Chose on their performance...

It is not just stacking learners on top of each other

- On has to be careful in how to set the estimate $\hat{x}_t$.
- Its rather a strategy learning...

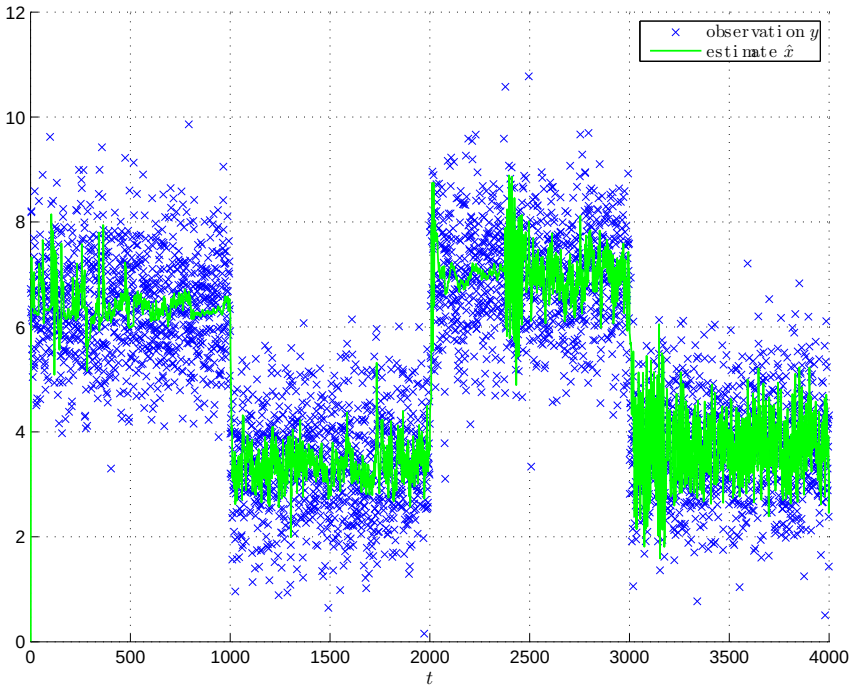
Further: Recursive parabola matching, multiple states Q-Learning

Example Run

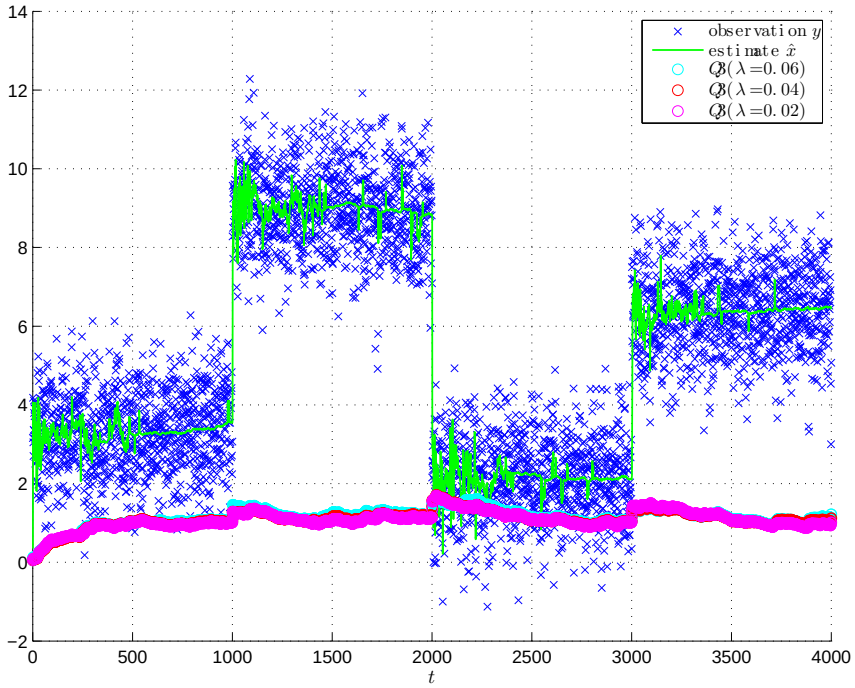


Tabu Learner

Example Run

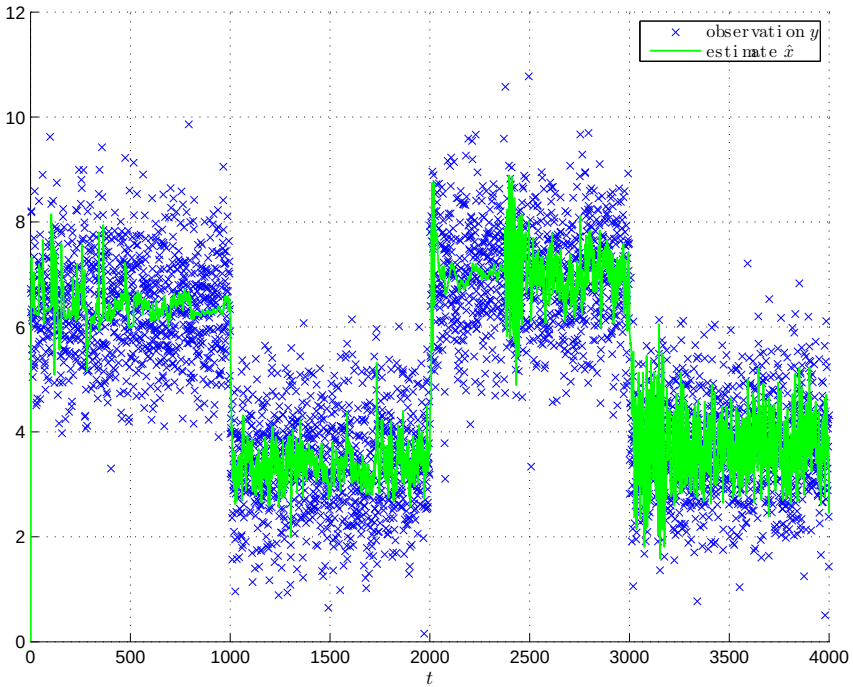


Tabu Learner

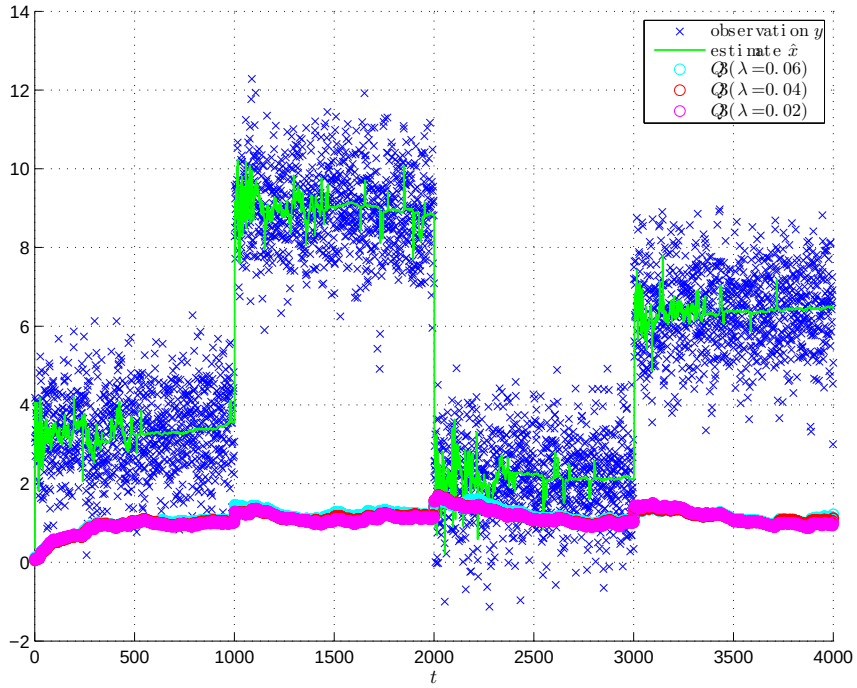


3-Layer Learner

Example Run



Tabu Learner



3-Layer Learner

Method	Parameters		mean$(x-\hat{x})^2$		
	$\lambda 2$	$\lambda 3$	C=2	C=4	C=20
Tabu policy single learner			0.132	0.182	0.607
2-Layer	0.06		0.067 (50.7%)	0.084 (46.3%)	0.203 (33.5%)
	0.04		0.053 (40.5%)	0.074 (40.8%)	0.201 (**33.1%**)
	0.02		0.044 (**33.5%**)	0.069 (**37.9%**)	0.218 (36.0%)
3-Layer	0.02, 0.04, 0.06	0.005	0.044 (33.1%)	0.071 (38.9%)	0.204 (33.6%)

The three layer method combines the good performance of different $\lambda 2$ learners.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

  Task What is a good task?
  $\Rightarrow$ Explore other tasks.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

On initial Task

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

$\Rightarrow$ Use a different learning rate set

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

$\Rightarrow$ Use a different learning rate set

! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

$\Rightarrow$ Use a different learning rate set

! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

- Exploration on task

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

$\Rightarrow$ Use a different learning rate set

! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook ($\Rightarrow$)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. $\longrightarrow$ Questions the task and the method.

Task	What is a good task?
$\Rightarrow$	Explore other tasks.
Method	Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
$\Rightarrow$	Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

- Exploration on task
- Exploration on concurrent learners

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

$\Rightarrow$ Use a different learning rate set

- ! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook (⇒)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T (x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T (u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. ⟶ Questions the task and the method.

  Task: What is a good task?
  ⇒ Explore other tasks.
  Method: Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
  ⇒ Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

- Exploration on task
- Exploration on concurrent learners
  - \+ 3-Layer Learner

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

⇒ Use a different learning rate set

- ! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook (⇒)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. ⟶ Questions the task and the method.

  Task: What is a good task?
  ⇒ Explore other tasks.
  Method: Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
  ⇒ Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

- Exploration on task
- Exploration on concurrent learners
  - \+ 3-Layer Learner
  - ⇒ Multi-Layer learner, other discounted recursive learners, multi state learners

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

⇒ Use a different learning rate set

- ! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Conclusion and Outlook (⇒)

For using PRL to compare machine to human according to risk and volatility

- \+ Task: λ_{opt}(Volatility, Risk) (opposite as proposed in paper)
- \+ RLLR: η^2 as a feedback works well in a $\frac{1}{T}\sum_1^T(x_t - \hat{x}_t)^2$ sense.
- – RLLR: η^2 as a feedback works partially well in a $\frac{1}{T}\sum_1^T(u_t \neq \hat{u}_t)$ (Maximizing Reward Error) sense. ⟶ Questions the task and the method.

  Task: What is a good task?
  ⇒ Explore other tasks.
  Method: Is discounted recursive mean learning the right choice for learning $\hat{x}$? Is η^2 the right feedback to use when using a λ_{opt} learning approach?
  ⇒ Explore on using $(\hat{u}_{t-1} = y_t)$ as a feedback for the learning rate learner.

Additional Contribution

- Exploration on task
- Exploration on concurrent learners
  - \+ 3-Layer Learner
  - ⇒ Multi-Layer learner, other discounted recursive learners, multi state learners

On initial Task

- ! Initial Q-values are crucial for successful learning.
  - \+ We gave bounds to estimate.
  - – In original paper too low chosen.
- ! Properties of a good learner:
  1. Avoids violating Markov property
  2. Policy considers convexity of λ_{opt}
- Tabu Policy:
  - \+ Profits from neighborhood exploration
  - – No profit from banning
- ! Q-Learning
  - – No advantage of Q-Learning over recursive discounted mean calculation.
  - Learning rate ($\lambda 2$) depends on dynamics of λ_{opt}.

⇒ Use a different learning rate set

! 2: Concurrent learners are superior. In static setting they are optimal, hence it is important to use dynamic task according to risk and volatility.

Q-Learning the Learning Rate

⇒ We did not focus further on this Idea. But we assume potential when using in a multi-state setup.

Questions?