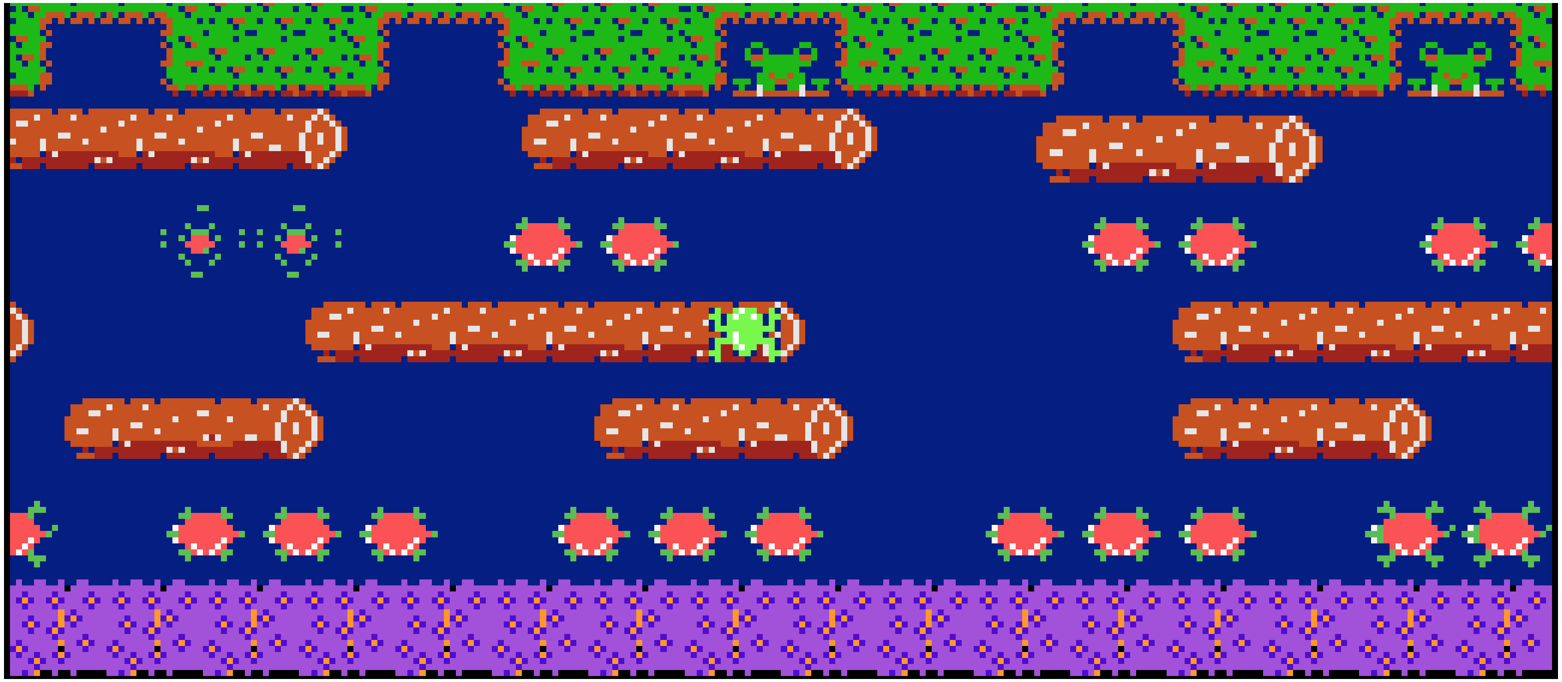
ONLINE LEARNING IN MARKOV DECISION PROCESSES

Gergely Neu
INRIA Lille, Sequel
Joint work with Alexander Zimin, Csaba Szepesvári and András György

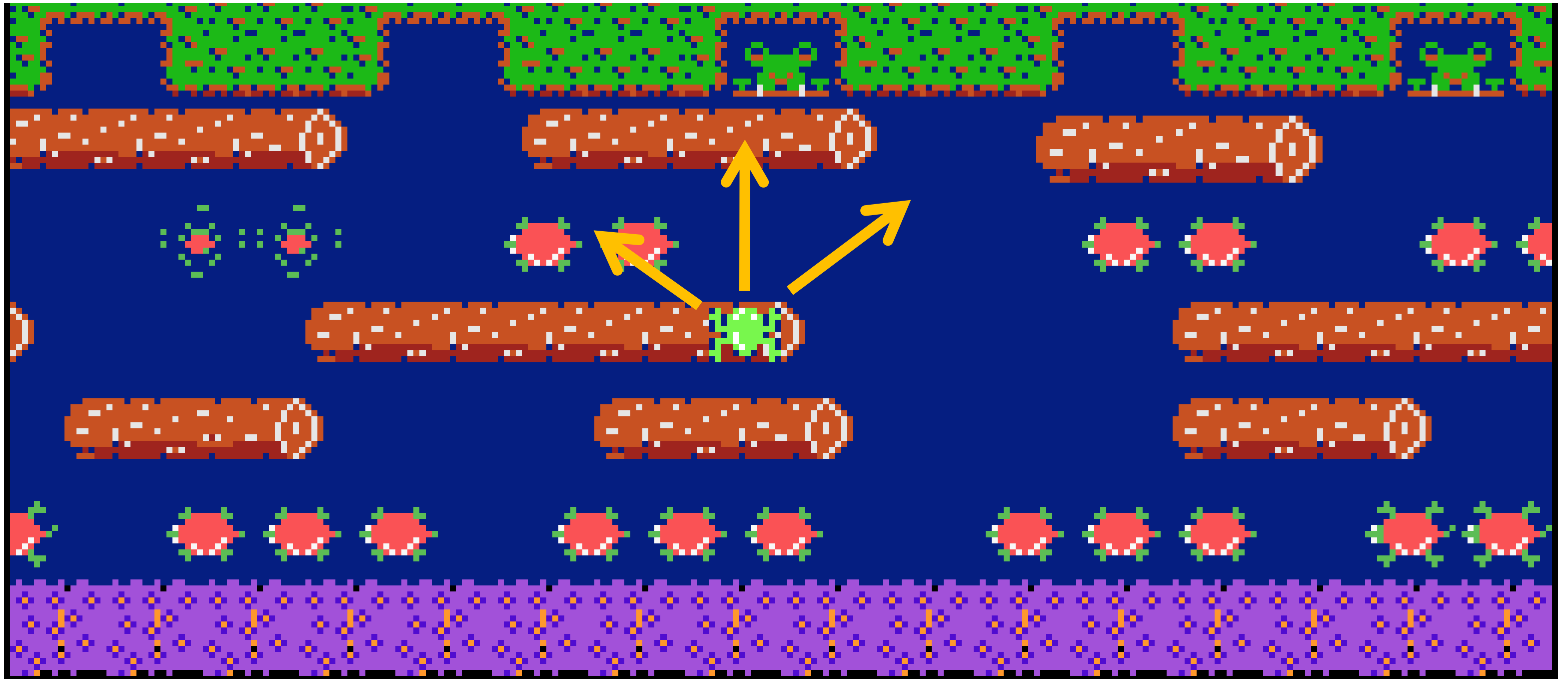
OUTLINE

1. The learning model
2. Regret
3. A simple algorithm: MDP-EXP3
4. A near-optimal algorithm: Relative Entropy Policy Search
5. Conclusions

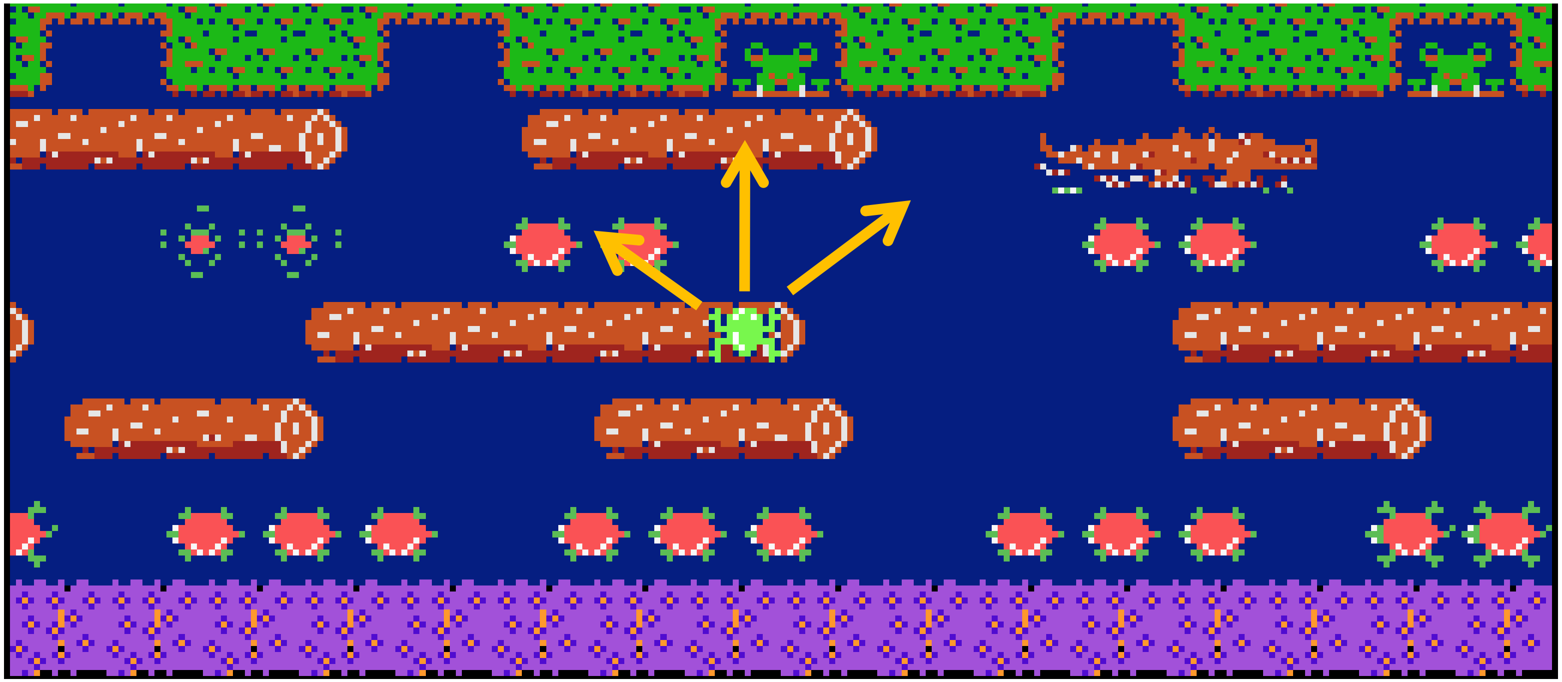
AN EXAMPLE



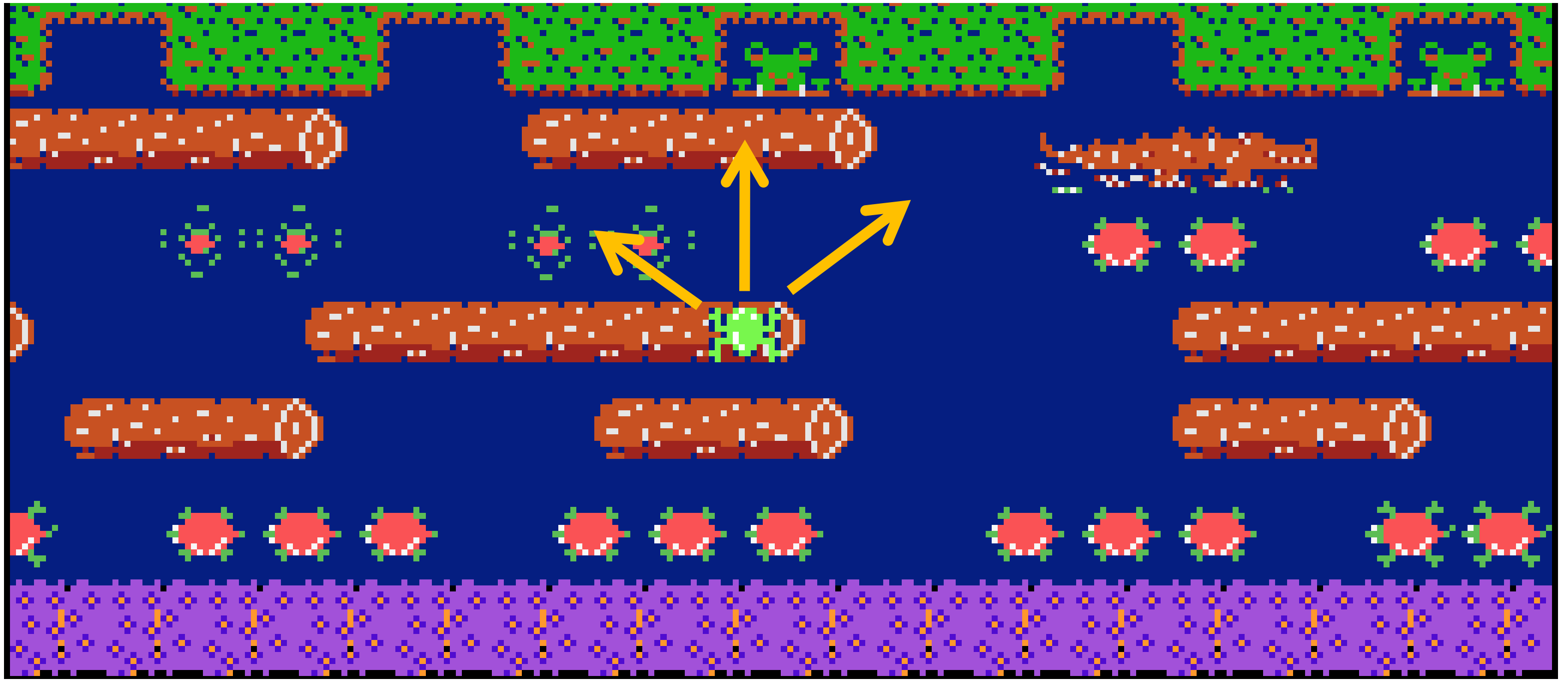
AN EXAMPLE



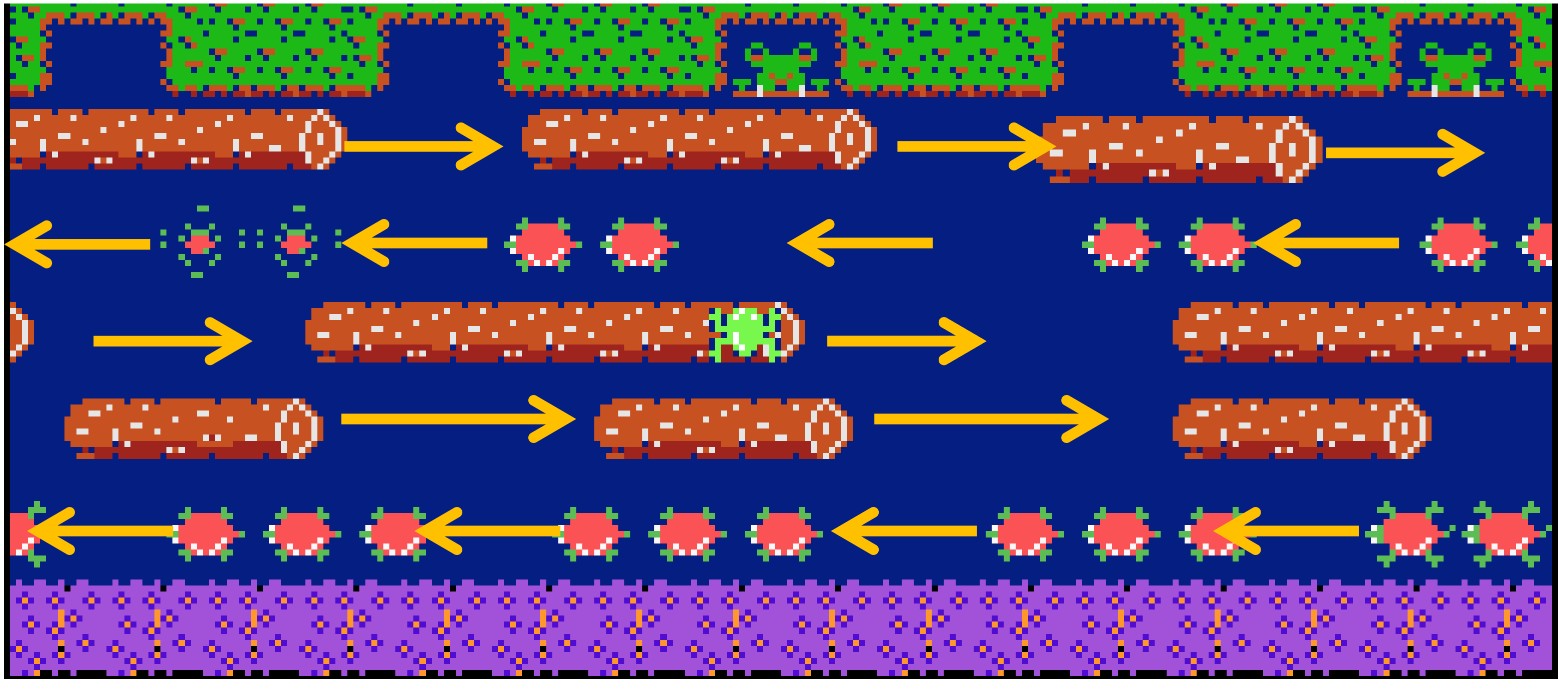
AN EXAMPLE



AN EXAMPLE

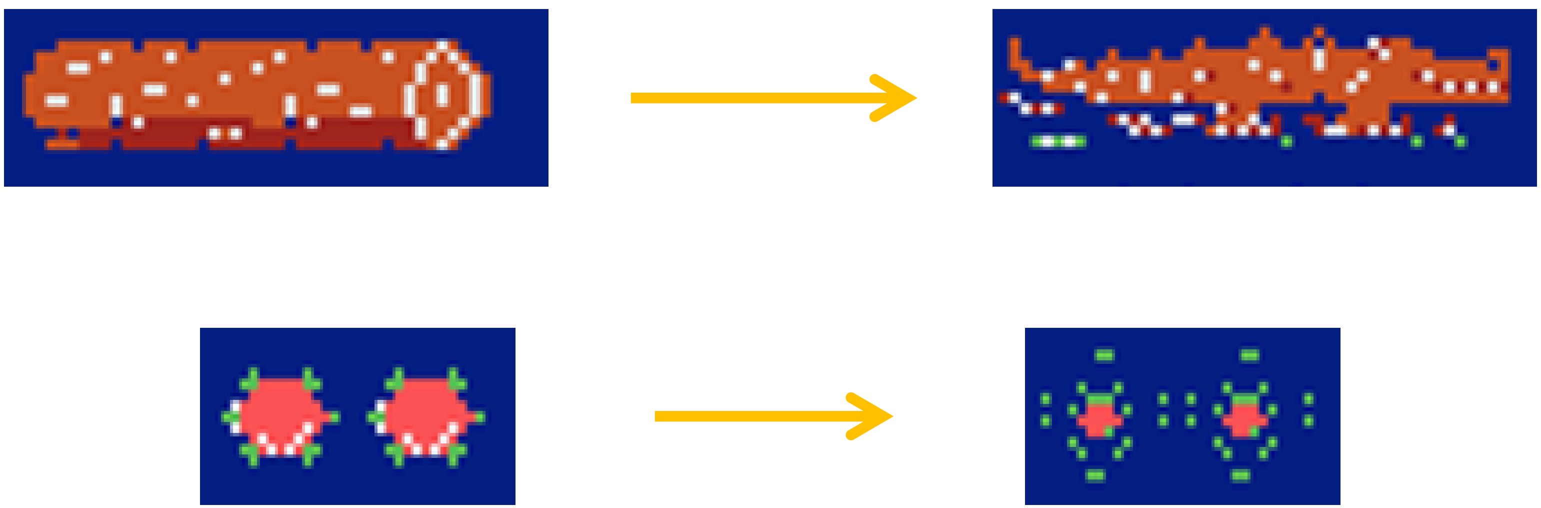


AN EXAMPLE: EASY PART



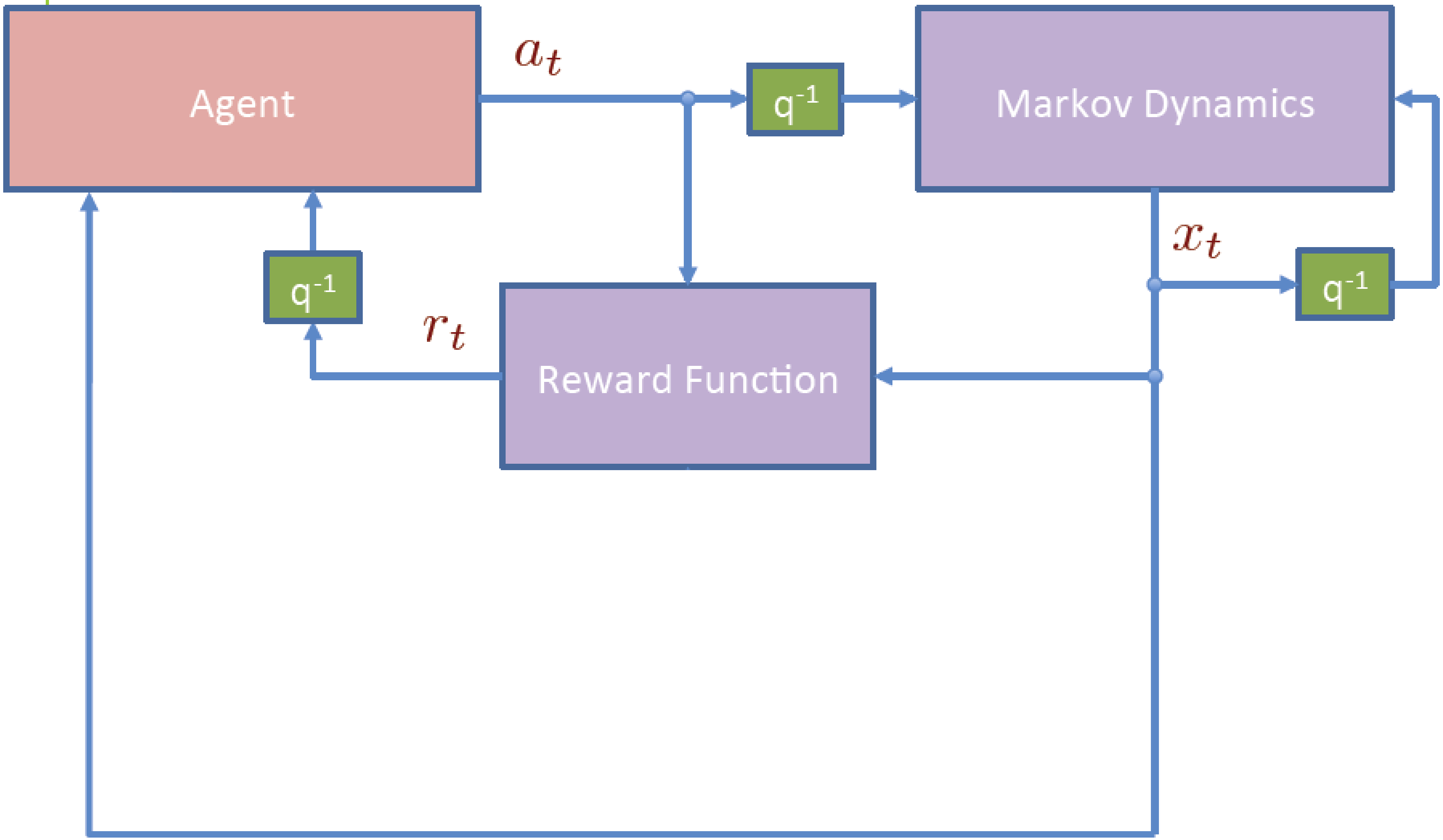
$\sim$ Simple random dynamics

AN EXAMPLE: DIFFICULT PART

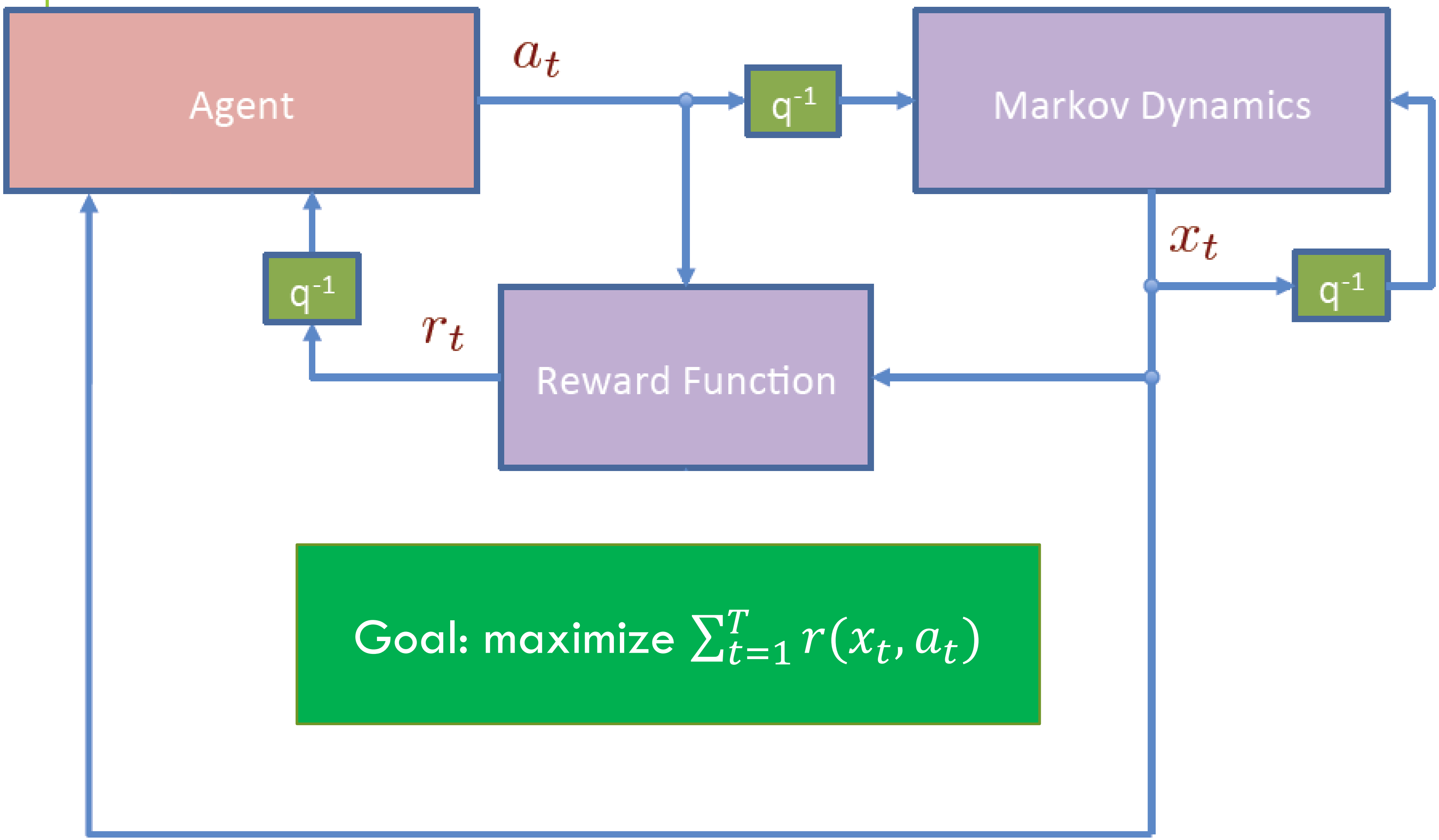


~ Nontrivial dynamics

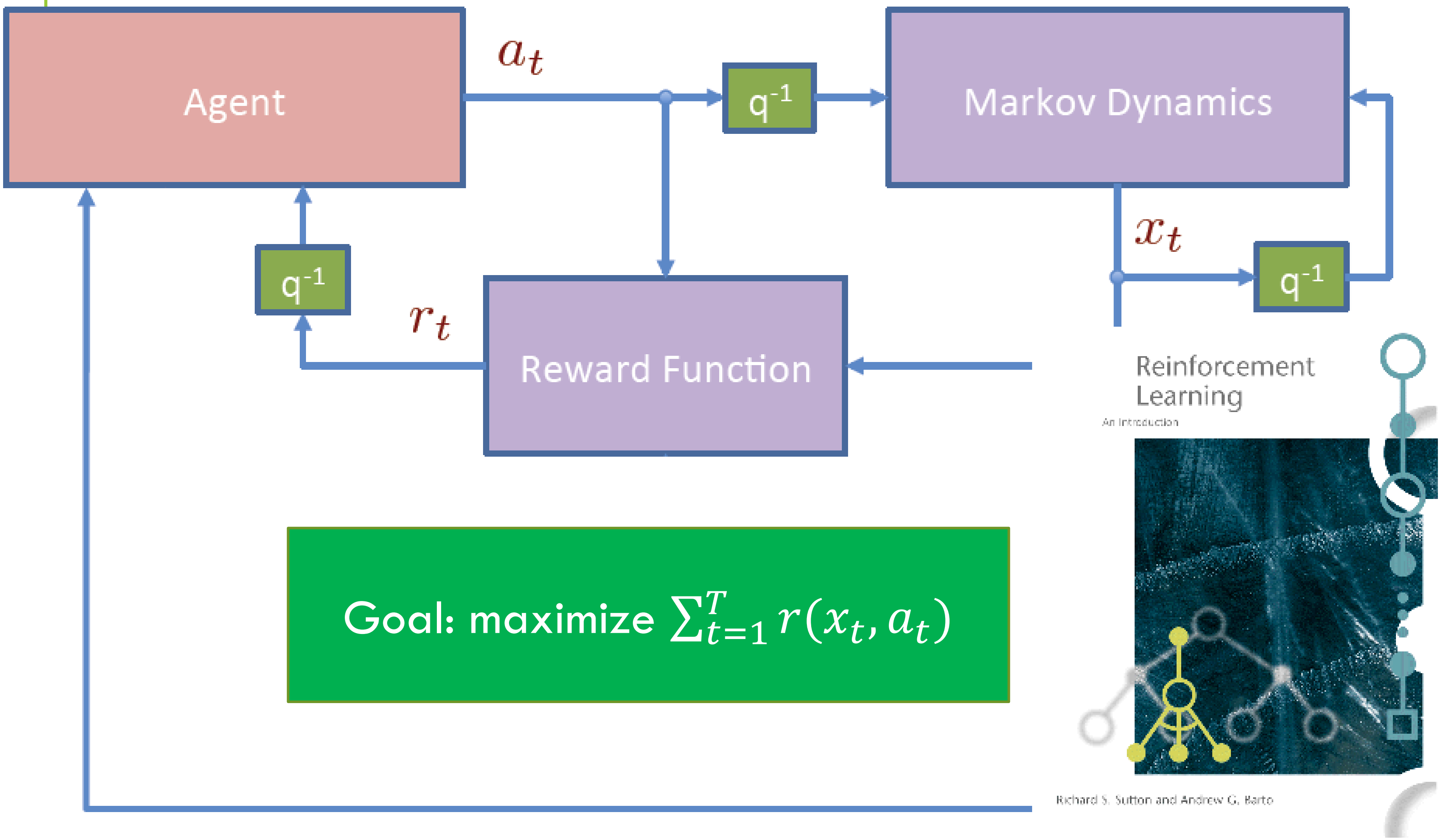
MARKOV DECISION PROCESSES



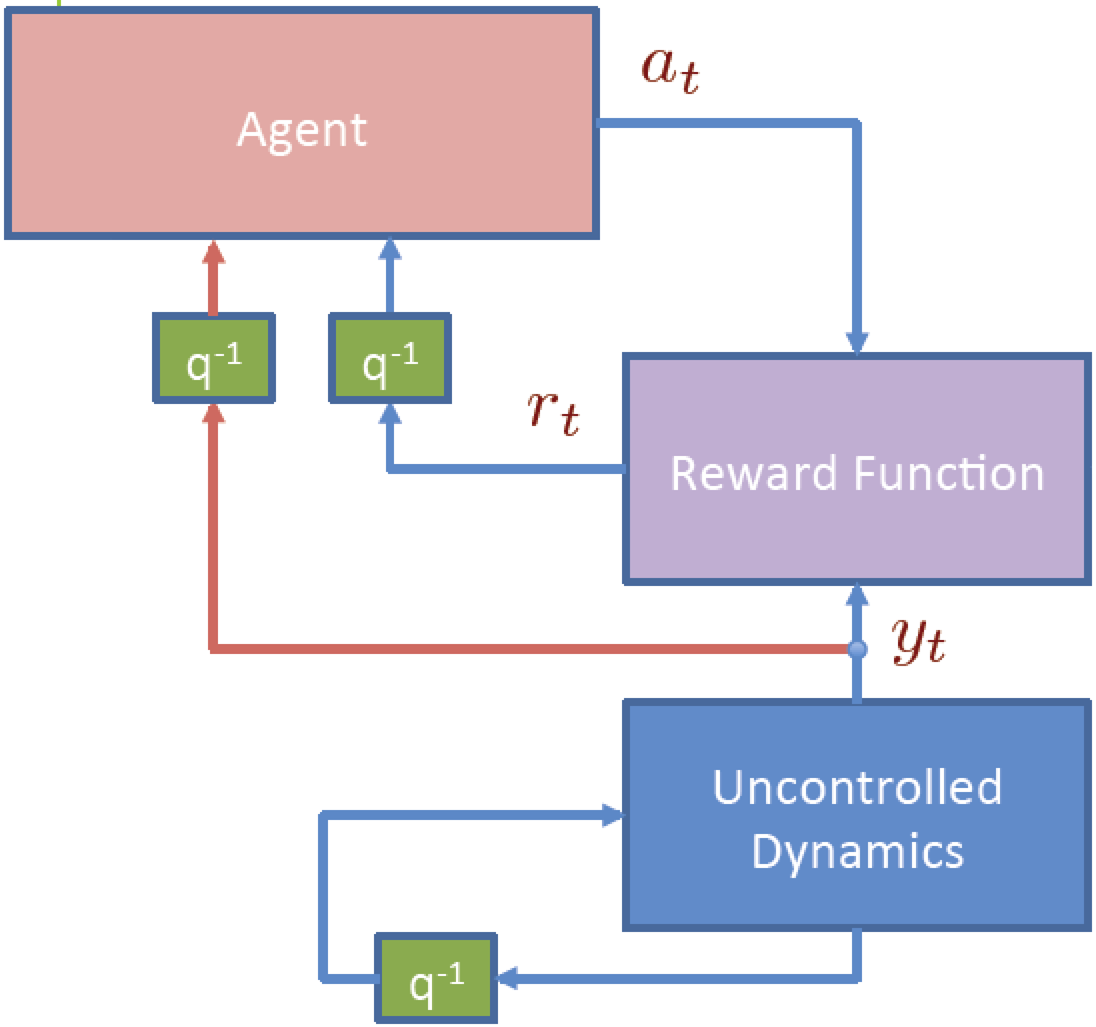
MARKOV DECISION PROCESSES



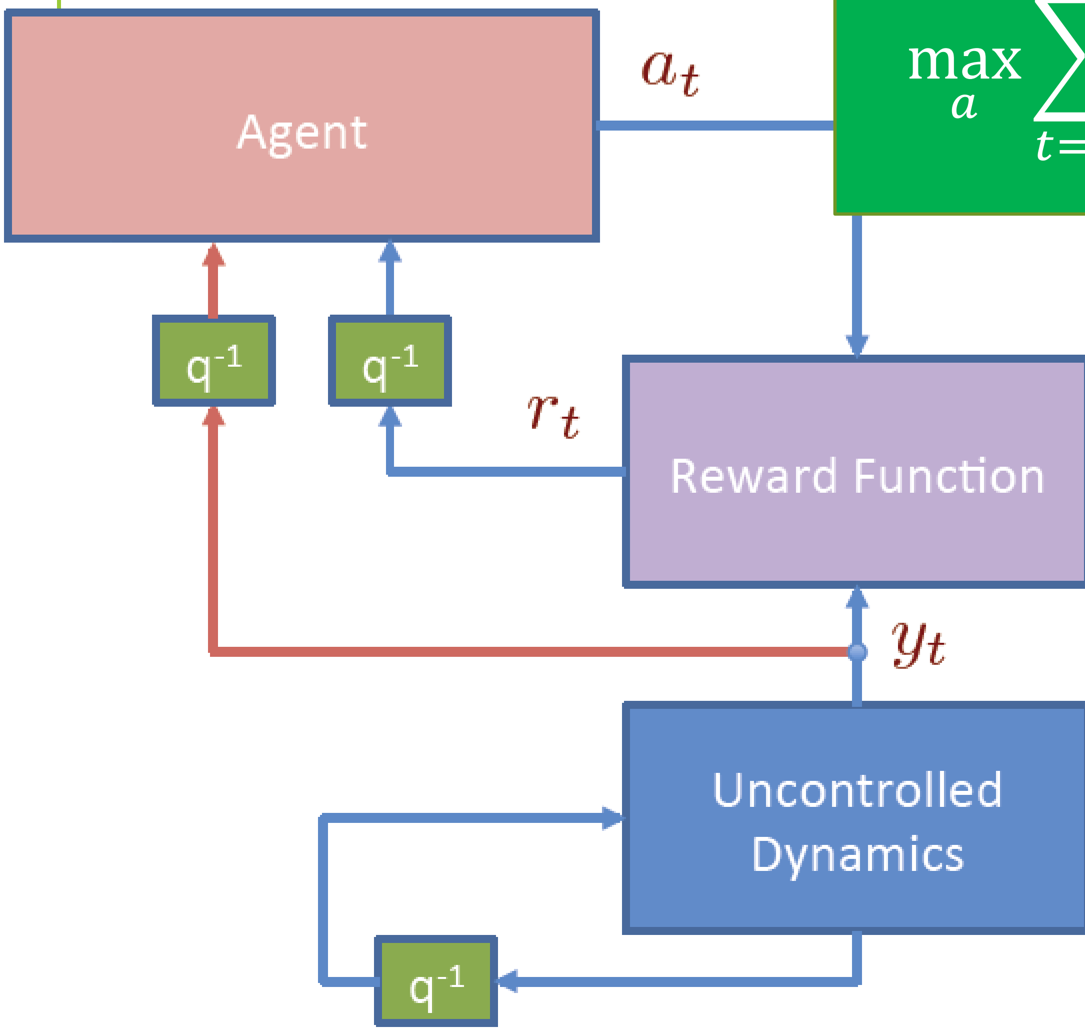
MARKOV DECISION PROCESSES



ONLINE LEARNING



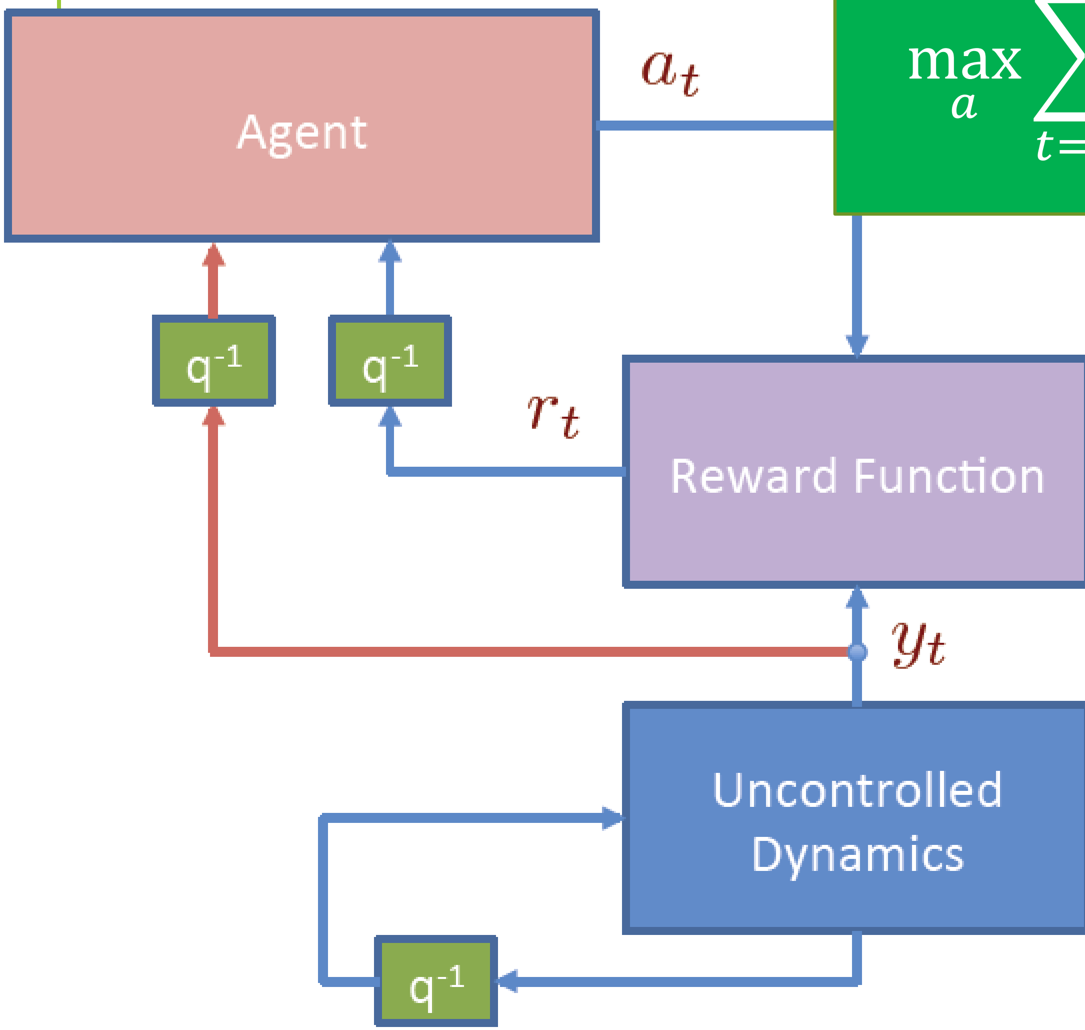
ONLINE LEARNING



Goal: minimize regret

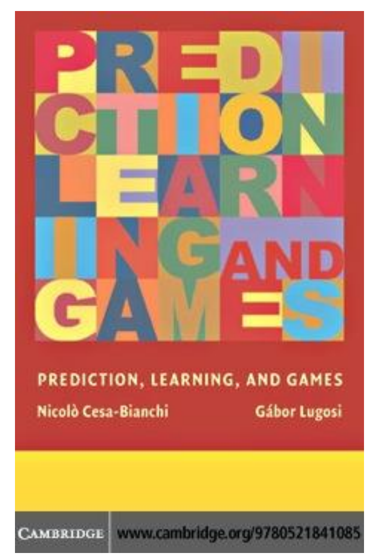
$$\max_a \sum_{t=1}^{T} r(y_t, a) - \sum_{t=1}^{T} r(y_t, \mathbf{a}_t)$$

ONLINE LEARNING

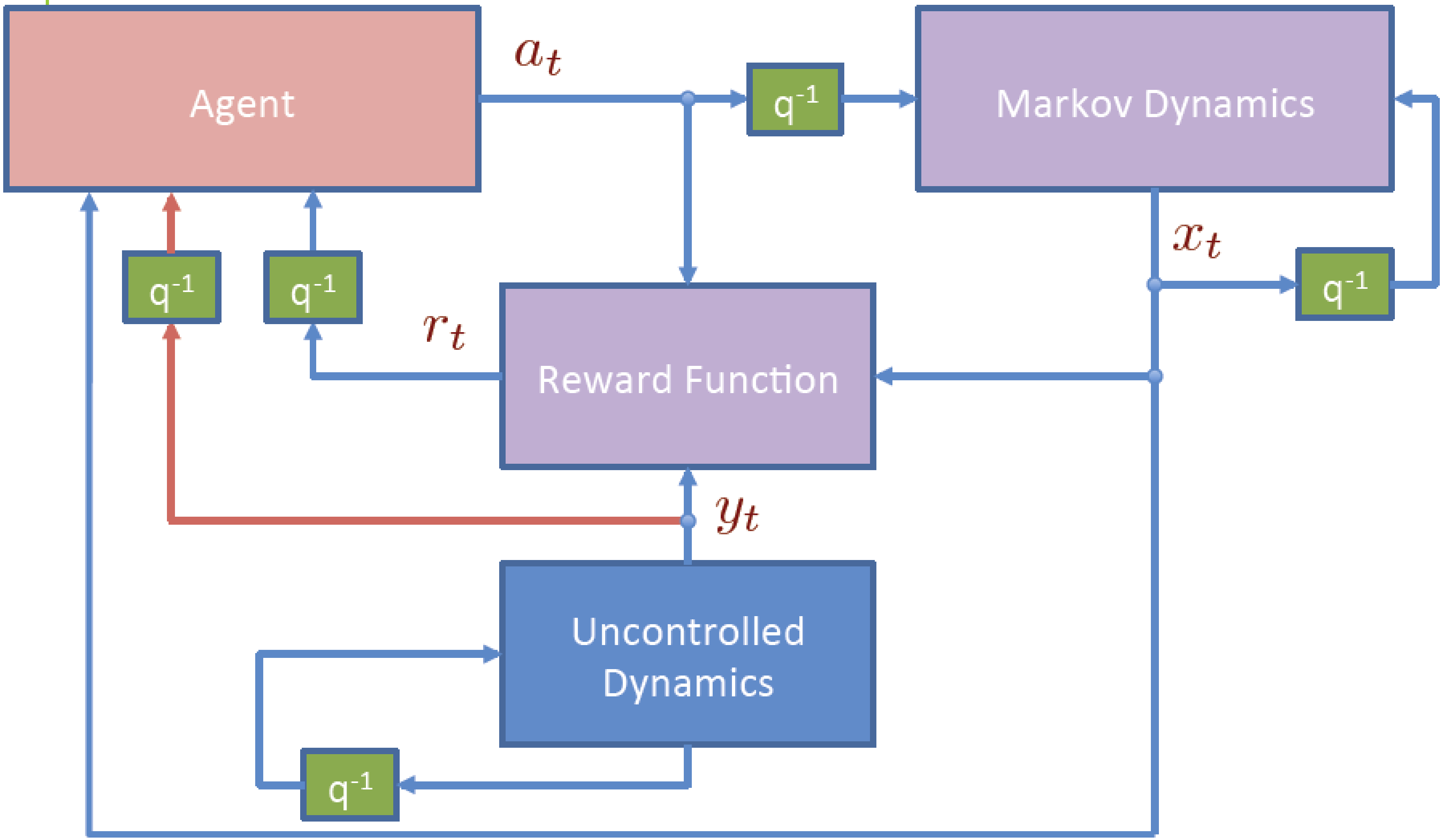


Goal: minimize regret

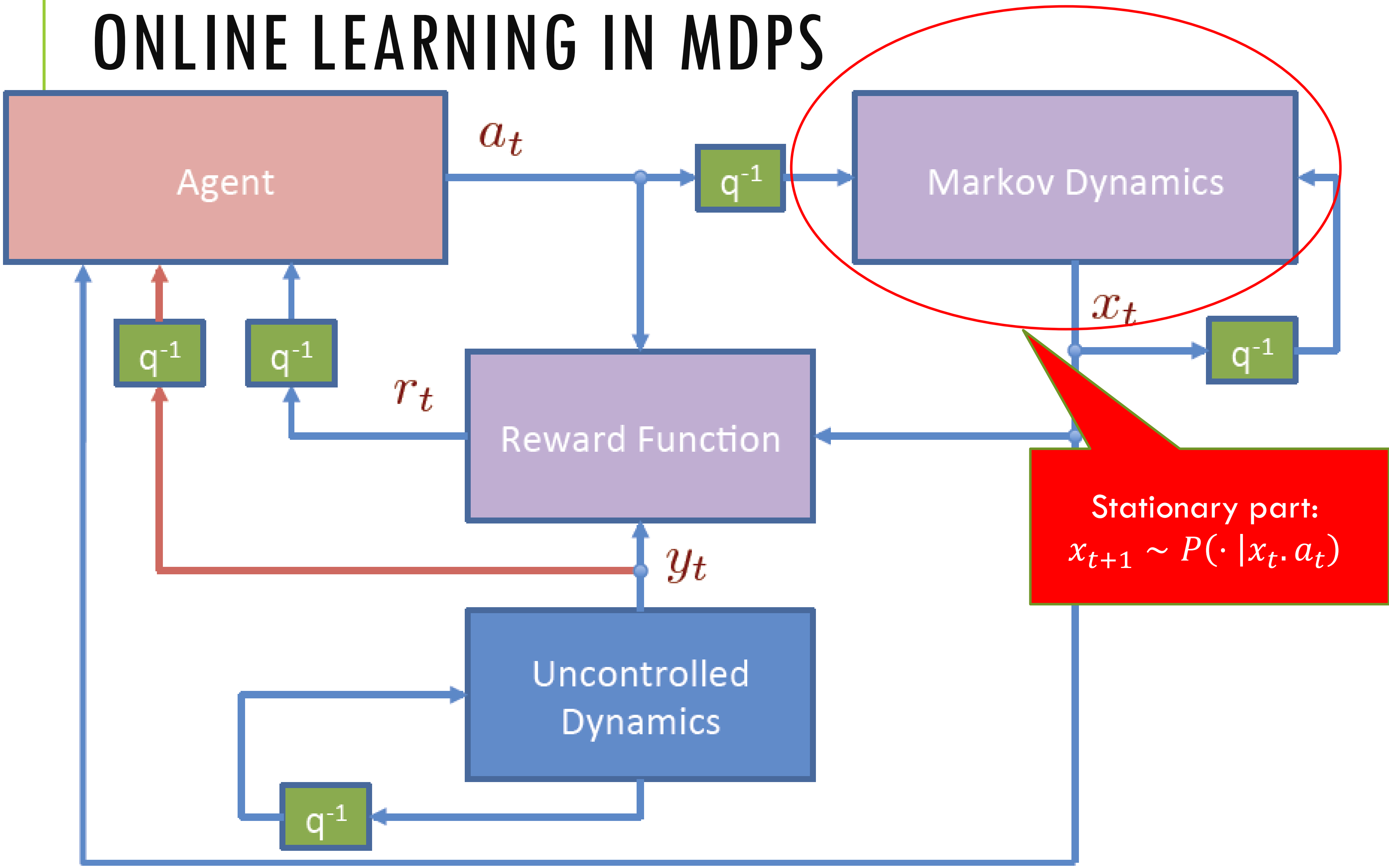
$$\max_a \sum_{t=1}^{T} r(y_t, a) - \sum_{t=1}^{T} r(y_t, \mathbf{a}_t)$$



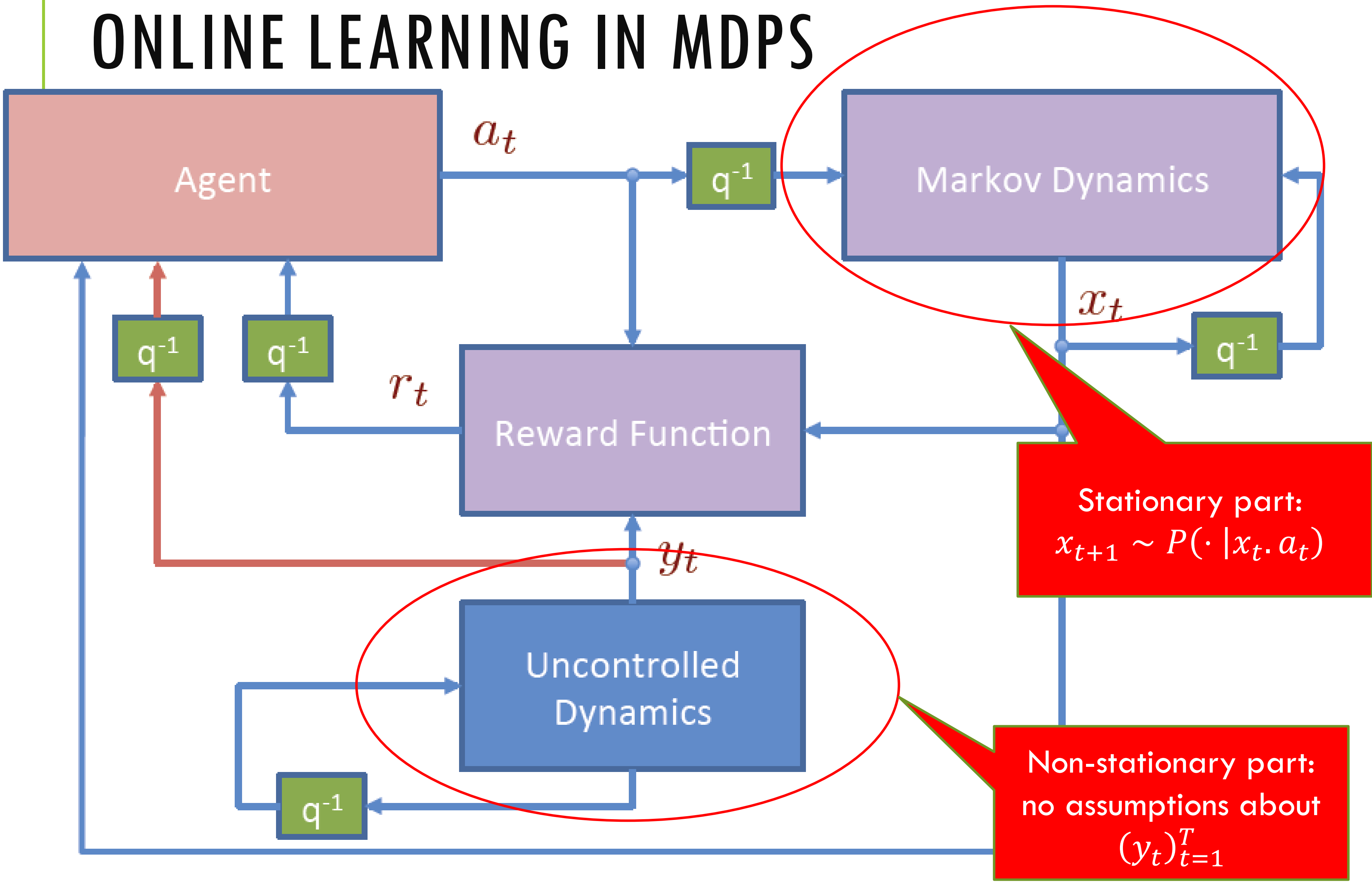
ONLINE LEARNING IN MDPS



ONLINE LEARNING IN MDPS



ONLINE LEARNING IN MDPS



SOME EXAMPLES

Sequential investment

- We influence positions, but not prices
- Prices effect revenue

SOME EXAMPLES

Sequential investment

- We influence positions, but not prices
- Prices effect revenue

Inventory management

Optimal control

Sequential routing

SOME EXAMPLES

Sequential investment

- We influence positions, but not prices
- Prices effect revenue

Inventory management

Optimal control

Sequential routing

Common factor

- Part of the state is controlled, with a well understood dynamics
- Part of the state is uncontrolled, complicated dynamics, unobserved state variable
- Only the reward is influenced by the uncontrolled component

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \rightarrow [0,1]$: known model of controlled states

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \to [0,1]$: known model of controlled states

$P(x'|x,a)$ is the probability of moving to state x' when choosing action a in state x

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \to [0,1]$: known model of controlled states

$P(x'|x,a)$ is the probability of moving to state x' when choosing action a in state x

$r_t: X \times A \to [0,1]$: reward function in episode t

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \to [0,1]$: known model of controlled states

$P(x'|x,a)$ is the probability of moving to state x' when choosing action a in state x

$r_t: X \times A \to [0,1]$: reward function in episode t

$r_t(x,a)$ is the reward given for choosing action a in state x in episode t

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \to [0,1]$: known model of controlled states

$P(x'|x, a)$ is the probability of moving to state x' when choosing action a in state x

$r_t: X \times A \to [0,1]$: reward function in episode t

$r_t(x, a)$ is the reward given for choosing action a in state x in episode t

$\pi: A \times X \to [0,1]$: policy

NOTATION

X: finite set of states of controlled dynamics (state space)

$A = \cup_{x \in X} A(x)$: finite action space

$P: X \times X \times A \to [0,1]$: known model of controlled states

$P(x'|x,a)$ is the probability of moving to state x' when choosing action a in state x

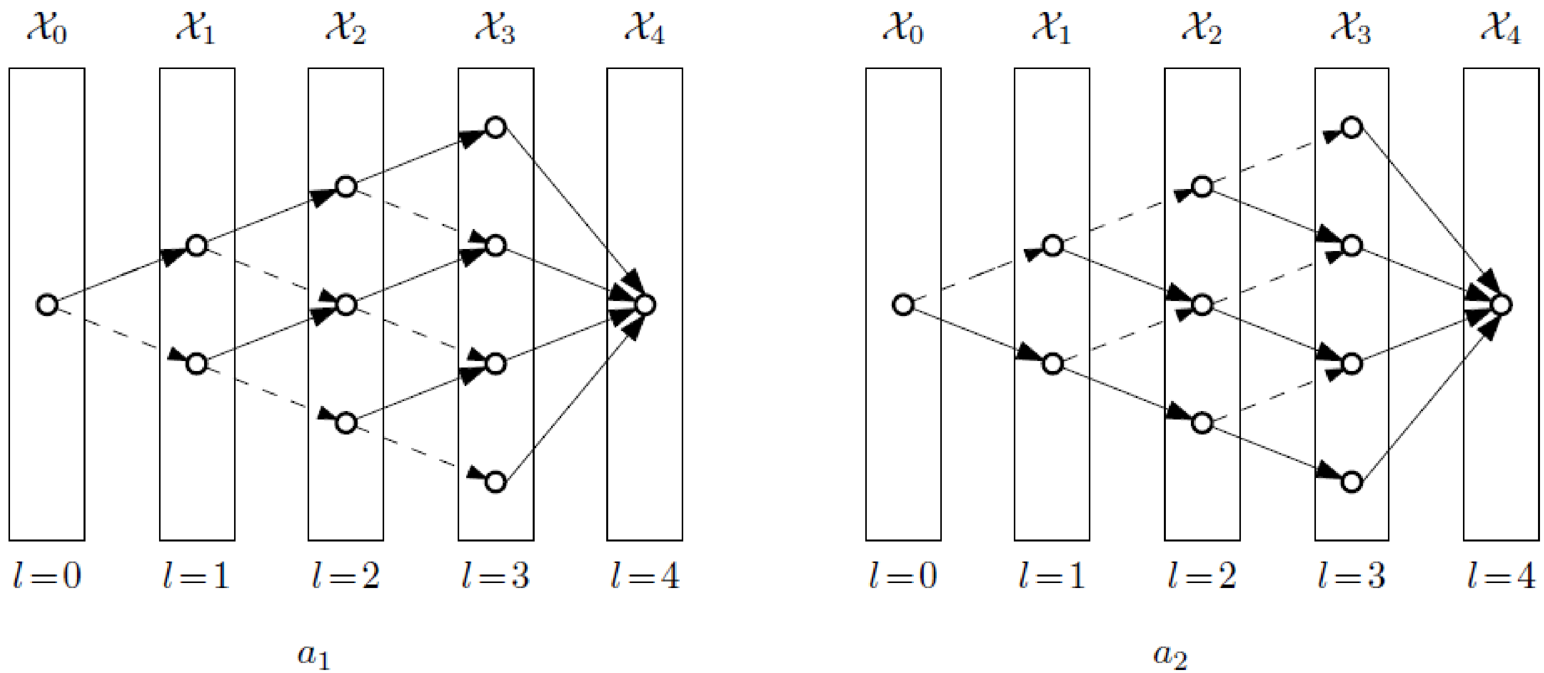
$r_t: X \times A \to [0,1]$: reward function in episode t

$r_t(x,a)$ is the reward given for choosing action a in state x in episode t

$\pi: A \times X \to [0,1]$: policy

$\pi(a|x)$ is the probability of choosing action a in state x

LOOP-FREE EPISODIC MDPS



Number of layers: L

ONLINE LEARNING IN EPISODIC MDPS

- For each episode $t = 1,2, \dots, T$
  - Learner chooses policy π_t
  - Adversary selects rewards $\boldsymbol{r}_t \in [0,1]^{X \times A}$
  - Learner traverses path $\boldsymbol{u}_t \sim (\pi_t, P)$
  - Learner gains $\langle \boldsymbol{u}_t, \boldsymbol{r}_t \rangle$
  - Based on $\boldsymbol{u}_t$ and $\boldsymbol{r}_t$, the learner gets some feedback

ONLINE LEARNING IN EPISODIC MDPS

- For each episode $t = 1,2, \dots, T$
  - Learner chooses policy π_t
  - Adversary selects rewards $\boldsymbol{r}_t \in [0,1]^{X \times A}$
  - Learner traverses path $\boldsymbol{u}_t \sim (\pi_t, P)$
  - Learner gains $\langle \boldsymbol{u}_t, \boldsymbol{r}_t \rangle$
  - Based on $\boldsymbol{u}_t$ and $\boldsymbol{r}_t$, the learner gets some feedback

Full info: $\boldsymbol{r}_t$

Bandit info: $r_t(x, a)$ for all $(x, a) \in \boldsymbol{u}_t$

REGRET

Let

$$\rho_t^\pi = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}\left[\sum_{x,a} u(x,a) r_t(x,a)\right] = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}[\langle \boldsymbol{u}, \boldsymbol{r}_t \rangle]$$

REGRET

Let

$$\rho_t^\pi = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}\left[\sum_{x,a} u(x,a) r_t(x,a)\right] = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}[\langle \boldsymbol{u}, \boldsymbol{r}_t \rangle]$$

Goal: choose policies $\pi_1, \pi_2, \dots, \pi_T$ such that

$$\sum_{t=1}^{T} \rho_t^{\pi_t} \to \max$$

REGRET

Let

$$\rho_t^\pi = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}\left[\sum_{x,a} u(x,a) r_t(x,a)\right] = \mathbf{E}_{\boldsymbol{u}\sim(\pi,P)}[\langle \boldsymbol{u}, \boldsymbol{r}_t \rangle]$$

Goal: choose policies $\pi_1, \pi_2, \dots, \pi_T$ such that

$$\sum_{t=1}^{T} \rho_t^{\pi_t} \to \max$$

Performance is measured in terms of regret:

$$\boxed{\hat{L}_T = \max_\pi \sum_{t=1}^{T} \rho_t^\pi - \sum_{t=1}^{T} \rho_t^{\pi_t} \to \min}$$

A SOLUTION

Lemma (Neu, György and Szepesvári, 2010):

$$\hat{L}_T = \sum_x \mu^*(x) \sum_{t=1}^{T} \Big(Q_t\big(x, \pi_T^*(x)\big) - V_t(x) \Big)$$

A SOLUTION

Lemma (Neu, György and Szepesvári, 2010):

$$\hat{L}_T = \sum_x \mu^*(x) \underbrace{\sum_{t=1}^{T} \Big(Q_t\big(x, \pi_T^*(x)\big) - V_t(x) \Big)}_{\substack{\text{Regret in an online learning problem with} \\ \text{reward sequence } \{Q_t(x,\cdot)\}_{t=1}^{T}}}$$

A SOLUTION

Lemma (Neu, György and Szepesvári, 2010):

$$\hat{L}_T = \sum_x \mu^*(x) \underbrace{\sum_{t=1}^{T} \Big(Q_t\big(x, \pi_T^*(x)\big) - V_t(x) \Big)}$$

Regret in an online learning problem with reward sequence $\{Q_t(x,\cdot)\}_{t=1}^{T}$

Use an instance of a stateless bandit algorithm in all states $x \in X$!

MDP-EXP3

MDP-EXP3

In round t,

- Define action-value function

$$Q_t(x,a) = \mathbf{E}\left[\sum_{x',a'} u(x',a') r_t(x',a') \,\middle|\, \boldsymbol{u} \sim (\pi_t, P, (x,a))\right]$$

- For all states, define partition function

$$Z_t(x) = \sum_a \pi_t(a|x) e^{\eta Q_t(x,a)}$$

MDP-EXP3

In round t,

- Define action-value function

$$Q_t(x,a) = \mathbf{E}\left[\sum_{x',a'} u(x',a')r_t(x',a') \,\middle|\, \boldsymbol{u} \sim (\pi_t, P, (x,a))\right]$$

- For all … define partition function

$$\eta Q_t(x,a)$$

Update policy as

$$\pi_{t+1}(a|x) = \frac{\pi_t(a|x)e^{\eta Q_t(x,a)}}{Z_t(x)}$$

MDP-EXP3: GUARANTEES

Theorem 1 (Neu, György and Szepesvári, 2010):
Under full information, MDP-EXP3 satisfies

$$\hat{L}_T = O\left(L^2\sqrt{T\log|A|}\right)$$

Theorem 2 (Neu, György and Szepesvári, 2010):
Under bandit information, MDP-EXP3 satisfies

$$\hat{L}_T = O\left(L^2\sqrt{T|A|\log|A|/\alpha}\right)$$

MDP-EXP3: GUARANTEES

Theorem 1 (Neu, György and Szepesvári, 2010):
Under full information, MDP-EXP3 satisfies :'(

$$\hat{L}_T = O\left(L^2\sqrt{T\log|A|}\right)$$

Theorem 2 (Neu, György and Szepesvári, 2010):
Under bandit information, MDP-EXP3 satisfies :'(

$$\hat{L}_T = O\left(L^2\sqrt{T|A|\log|A|/\alpha}\right)$$

Decompose-then-bound
inevitably leads to loose bounds!

A GLOBAL SOLUTION

Average reward in episode t under π:

$$\rho_t^\pi = \mathbf{E}\left[\sum_{x,a} u(x,a) r_t(x,a) \middle| \boldsymbol{u} \sim (\pi, P)\right]$$

A GLOBAL SOLUTION

Average reward in episode t under π:

$$\rho_t^\pi = \mathbf{E}\left[\sum_{x,a} u(x,a) r_t(x,a) \middle| \boldsymbol{u} \sim (\pi, P)\right]$$

$$= \sum_{x,a} p^\pi(x,a) r_t(x,a) = \langle \boldsymbol{p}^\pi, \boldsymbol{r}_t \rangle$$

A GLOBAL SOLUTION

Average reward in episode t under π:

$$\rho_t^\pi = \mathbf{E}\left[\sum_{x,a} u(x,a) r_t(x,a) \middle| \boldsymbol{u} \sim (\pi, P)\right]$$

$$= \sum_{x,a} p^\pi(x,a) r_t(x,a) = \langle \boldsymbol{p}^\pi, \boldsymbol{r}_t \rangle$$

Rewards are linear in some representation!

THE STATE-ACTION POLYTOPE

Elements $\boldsymbol{p}$ of state-action polytope Δ satisfy:

$$\sum_{a} p(x,a) = \sum_{x',a'} P(x|x',a')p(x',a') \quad (\forall x)$$

$$\sum_{x \in X_k} \sum_{a} p(x,a) = 1 \quad (\forall k)$$

$$p(x,a) \geq 0 \quad (\forall x,a)$$

THE STATE-ACTION POLYTOPE

Elements $\boldsymbol{p}$ of state-action polytope Δ satisfy:

$$\sum_a p(x,a) = \sum_{x',a'} P(x|x',a')p(x',a') \quad (\forall x)$$

$$\sum_{x \in X_k} \sum_a p(x,a) = 1 \quad (\forall k)$$

$$p(x,a) \geq 0 \quad (\forall x,a)$$

Extracting policy:

$$\pi(a|x) = \frac{p(x,a)}{\sum_b p(x,b)}$$

ONLINE LEARNING IN EPISODIC MDPS

- For each episode $t = 1,2, \dots, T$
  - Learner chooses policy π_t
  - Adversary selects rewards $\boldsymbol{r}_t \in [0,1]^{X \times A}$
  - Learner traverses path $\boldsymbol{u}_t \sim (\pi_t, P)$
  - Learner gains $\langle \boldsymbol{u}_t, \boldsymbol{r}_t \rangle$
  - Based on $\boldsymbol{u}_t$ and $\boldsymbol{r}_t$, the learner gets some feedback

Full info: $\boldsymbol{r}_t$

Bandit info: $r_t(x, a)$ for all $(x, a) \in \boldsymbol{u}_t$

ONLINE LEARNING IN EPISODIC MDPS

- For each episode $t = 1,2, \dots, T$
  - Learner chooses distribution $\boldsymbol{p}_t \in \Delta$
  - Adversary selects rewards $\boldsymbol{r}_t \in [0,1]^{X \times A}$
  - Learner traverses path $\boldsymbol{u}_t \sim \boldsymbol{p}_t$
  - Learner gains $\langle \boldsymbol{u}_t, \boldsymbol{r}_t \rangle$
  - Based on $\boldsymbol{u}_t$ and $\boldsymbol{r}_t$, the learner gets some feedback

Full info: $\boldsymbol{r}_t$

Bandit info: $r_t(x, a)$ for all $(x, a) \in \boldsymbol{u}_t$

AN ALGORITHM: MIRROR DESCENT

Let $\boldsymbol{p}_1 \in \Delta$ and

$$\boxed{\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta}\left(-\eta\langle \boldsymbol{p}, \boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\right)}$$

AN ALGORITHM: MIRROR DESCENT

Let $\boldsymbol{p}_1 \in \Delta$ and

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta}\left(-\eta\langle\boldsymbol{p},\boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\right)$$

$$D(\boldsymbol{p}|\boldsymbol{q}) = \sum_{x,a} p(x,a)\log\frac{p(x,a)}{q(x,a)} - \sum_{x,a}\left(p(x,a) - q(x,a)\right)$$

GUARANTEES

Theorem 1 (Zimin and Neu, 2013, Dick et al. 2014):
Under full information, Mirror Descent satisfies

$$\hat{L}_T = O\left(L\sqrt{T\log|A|}\right)$$

Theorem 2 (Zimin and Neu, 2013, Dick et al. 2014):
Under bandit information, Mirror Descent satisfies

$$\hat{L}_T = O\left(\sqrt{L|X||A|T\log|A|}\right)$$

Proofs are similar to Koolen, Warmuth and Kivinen (2010), Audibert, Bubeck and Lugosi (2011,2014)

GUARANTEES

Theorem 1 (Zimin and Neu, 2013, Dick et al. 2014):
Under full information, Mirror Descent satisfies

$$\hat{L}_T = O\left(L\sqrt{T \log|A|}\right)$$

vs $O\left(L^2\sqrt{T \log|A|}\right)$

Theorem 2 (Zimin and Neu, 2013, Dick et al. 2014):
Under bandit information, Mirror Descent satisfies

$$\hat{L}_T = O\left(\sqrt{L|X||A|T \log|A|}\right)$$

vs $O\left(L^2\sqrt{T|A| \log|A| /\alpha}\right)$

Proofs are similar to Koolen, Warmuth and Kivinen (2010), Audibert, Bubeck and Lugosi (2011,2014)

"WHERE HAVE I SEEN THIS BEFORE?"

Mirror descent:

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta}\Big(-\eta\langle\boldsymbol{p},\boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\Big)$$

Relative Entropy Policy Search (Peters, Mülling, Altun, 2010):

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta}(-\langle\boldsymbol{p},\boldsymbol{r}_t\rangle)$$

s.t. $\quad D(\boldsymbol{p}|\boldsymbol{p}_t) \leq \varepsilon$

MIRROR DESCENT = ONLINE REPS

In round t,

- For a value function $v: X \to \mathbf{R}$, define Bellman error

$$\delta_t(x,a|v) = \eta r_t(x,a) + \sum_{x'} P(x'|x,a)v(x') - v(x)$$

- For all layers $k = 0,1,\ldots,L-1$, define partition function

$$Z_t(v,k) = \sum_{x \in X_k, a} p_t(x,a) e^{\delta_t(x,a|v)}$$

- Solve

$$V_t = \arg\min_v \sum_{k=0}^{L-1} \log Z_t(v,k)$$

MIRROR DESCENT = ONLINE REPS

In round t,

- For a value function $v: X \to \mathbf{R}$, define Bellman error

$$\delta_t(x, a|v) = \eta r_t(x, a) + \sum_{x'} P(x'|x, a)v(x') - v(x)$$

- For all lay $L - 1$, define partition function

$$p_{t+1}(x, a) = \frac{p_t(x, a)e^{\delta_t(x,a|V_t)}}{Z_t(V_t, k)}$$

- Solve

$$V_t = \arg\min_v \sum_{k=0}$$

DERIVATION OF THE UPDATE RULE

- Rewrite update in two steps:

$$\tilde{\boldsymbol{p}}_{t+1} = \arg \min_{\boldsymbol{p} \in R^{|X| \times |A|}} \left(-\eta \langle \boldsymbol{p}, \boldsymbol{r}_t \rangle + D(\boldsymbol{p} | \boldsymbol{p}_t) \right)$$

$$\boldsymbol{p}_{t+1} = \arg \min_{\boldsymbol{p} \in \Delta} D(\boldsymbol{p} | \tilde{\boldsymbol{p}}_{t+1})$$

DERIVATION OF THE UPDATE RULE

- Rewrite update in two steps:

$$\widetilde{\boldsymbol{p}}_{t+1} = \arg\min_{\boldsymbol{p}\in R^{|X|\times|A|}} \left(-\eta\langle \boldsymbol{p}, \boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\right)$$

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta} D(\boldsymbol{p}|\widetilde{\boldsymbol{p}}_{t+1})$$

- The first step is easy: $p_{t+1}(x,a) = p_t(x,a)e^{\eta r_t(x,a)}$

DERIVATION OF THE UPDATE RULE

- Rewrite update in two steps:

$$\widetilde{\boldsymbol{p}}_{t+1} = \arg\min_{\boldsymbol{p}\in R^{|X|\times|A|}} \left(-\eta\langle \boldsymbol{p}, \boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\right)$$

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta} D(\boldsymbol{p}|\widetilde{\boldsymbol{p}}_{t+1})$$

- The first step is easy: $p_{t+1}(x,a) = p_t(x,a)e^{\eta r_t(x,a)}$
- The projection step is a constrained optimization problem with equality constraints → Lagrange multipliers:
  - $v(x)$ for flow constraints
  - $Z_t(v,k)$ for normalization constraints

DERIVATION OF THE UPDATE RULE

- Rewrite update in two steps:

$$\widetilde{\boldsymbol{p}}_{t+1} = \arg\min_{\boldsymbol{p}\in R^{|X|\times|A|}} \left(-\eta\langle\boldsymbol{p},\boldsymbol{r}_t\rangle + D(\boldsymbol{p}|\boldsymbol{p}_t)\right)$$

$$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\Delta} D(\boldsymbol{p}|\widetilde{\boldsymbol{p}}_{t+1})$$

- The first step is easy: $p_{t+1}(x,a) = p_t(x,a)e^{\eta r_t(x,a)}$
- The projection step is a constrained optimization problem with equality constraints → Lagrange multipliers:
  - $v(x)$ for flow constraints
  - $Z_t(v,k)$ for normalization constraints

"Value function" comes from solving the dual

MDP-EXP3 VS O-REPS

	MDP-EXP3	O-REPS
Value function	Solve Bellman-eq. (Global)	Solve dual (Global)
Update rule	$\pi_t(a\|x)e^{\eta Q_t(x,a)}$ (Local)	$p_t(x,a)e^{\delta_t(x,a\|V_t)}$ (Global)
Normalization	Per state (Local)	Per layer (Global)
Guarantees	$\hat{L}_T(x) = \tilde{O}(L\sqrt{T})$ (Per state, local)	$\hat{L}_T = \tilde{O}(L\sqrt{T})$ (Global)

WHAT'S THE LESSON?

Suboptimal ideas:

- Decomposition
- Sticking to traditional Bellman-equations

WHAT'S THE LESSON?

Suboptimal ideas:

- Decomposition
- Sticking to traditional Bellman-equations

Good ideas:

- Using the LP formulation
- Regularizing with relative entropy

OPEN PROBLEMS & FUTURE DIRECTIONS

- Computing the O-REPS updates
  - Needs solving an unconstrained convex program
  - Might be solvable by dynamic programming (Gerhard Neumann, p.c.)

OPEN PROBLEMS & FUTURE DIRECTIONS

- Computing the O-REPS updates
  - Needs solving an unconstrained convex program
  - Might be solvable by dynamic programming (Gerhard Neumann, p.c.)
- Analyzing the original REPS
  - Parameter tuning seems easier
  - Standard analysis tools no longer apply

OPEN PROBLEMS & FUTURE DIRECTIONS

- Computing the O-REPS updates
  - Needs solving an unconstrained convex program
  - Might be solvable by dynamic programming (Gerhard Neumann, p.c.)
- Analyzing the original REPS
  - Parameter tuning seems easier
  - Standard analysis tools no longer apply
- Scaling it up to large/continuous state spaces
  - Approximate updates or feature-based REPS

THANKS!