# Pump-Priming Projects “Online Performance of Reinforcement Learning” and “Sequential Forecasting and Partial Feedback”

Peter Auer

University of Leoben, Austria

Bled, 29 January 2008

# Online performance of reinforcement learning with internal reward functions

# Online performance of reinforcement learning with internal reward functions

Proposers:

# Online performance of reinforcement learning with internal reward functions

Proposers:

- John Shawe-Taylor, University College London

# Online performance of reinforcement learning with internal reward functions

Proposers:

- Bernhard Schölkopf, MPI Tübingen
- John Shawe-Taylor, University College London

# Online performance of reinforcement learning with internal reward functions

Proposers:

- Peter Auer, University of Leoben
- Bernhard Schölkopf, MPI Tübingen
- John Shawe-Taylor, University College London

# Online performance of reinforcement learning with internal reward functions

Proposers:

- Peter Auer, University of Leoben
- Bernhard Schölkopf, MPI Tübingen
- John Shawe-Taylor, University College London

Goals:

- Online analysis of reinforcement learning
- Online analysis for continuous state spaces
- Design of internal reward functions

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Sequential Forecasting and Partial Feedback: Applications to Machine Learning

Proposers:

- Peter Auer (Austria), Nicolò Cesa-Bianchi (Italy), Claudio Gentile (Italy), András György (Hungary), Gábor Lugosi (Spain), Yishay Mansour (Israel), Csaba Szepesvári (Canada)

Goals:

- Use machine learning techniques for parameter tuning
- Use inverse reinforcement learning for apprenticeship learning
- Sequential forecasting when the target (e.g. user interest) is changing

# Activities (partial list)

- Hired a post doctoral researcher (Christos Dimitrakakis) and a PhD student (Ivett Szabó).
- Organized the PASCAL workshop "Principled methods of trading exploration and exploitation Workshop" in London.
- Organized PASCAL "Exploration vs. Exploitation Challenge".
- Organized NIPS workshop "On-line trading of Exploration and Exploitation Workshop" in Canada.
- Organized a workshop on reinforcement learning in Tübingen. Remi Munos took the initiative to reestablish the **European Workshop on Reinforcement Learning**, Lille 2008.
- Expertise from these projects is used in a new **7th framework STREP: PinView** (Personal Information Navigator Adapting Through Viewing).

# Scientific outcome (partial list)

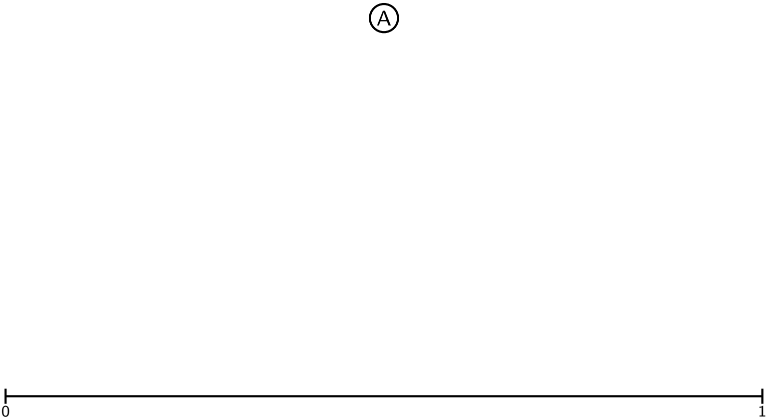
- P. Auer and R. Ortner: Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning, **NIPS 2006**.
- P. Auer, R. Ortner, and C. Szepesvari: Improved Rates for the Stochastic Continuum-Armed Bandit Problem, **COLT 2007**.
- G. Neu and Cs. Szepesvári: Apprenticeship learning using inverse reinforcement learning and gradient methods, **UAI 2007**.
- A. György, T. Linder, G. Lugosi, and Gy. Ottucsák: The on-line shortest path problem under partial monitoring, **JMLR 2007**.
- R. Ortner: Linear Dependence of Stationary Distributions in Ergodic Markov Decision Processes, **OR Letters 2007**.
- R. Ortner: Pseudometrics for State Aggregation in Average Reward Markov Decision Processes, **ALT 2007**.
- Ch. Dimitrakakis and Ch. Savu-Krohn: Cost-minimising strategies for data labelling - optimal stopping and active learning, **FoIKS 2008**.
- P. Auer, R. Ortner, T. Jaksch: Near-optimal Regret Bounds for Reinforcement Learning, submitted.

# ML algorithms for parameter optimization: UCT

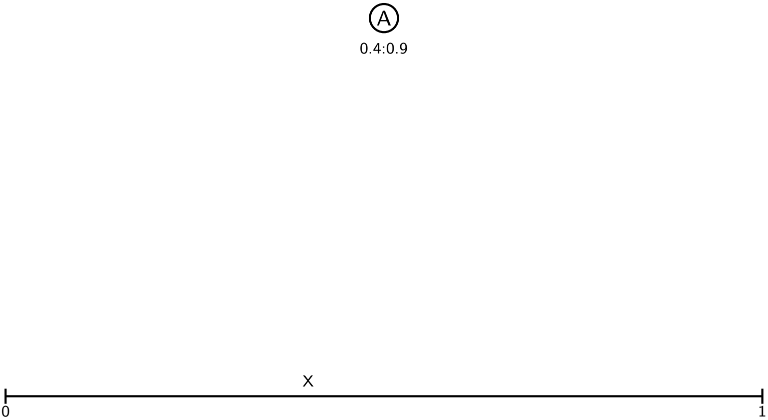
- The UCT (upper confidence for trees) algorithm [KS 2006] is a method for exploring trees, based on the UCB algorithm for the bandit problem.
- Used also in MoGo (world champion in computer Go, Sylvain Gelly et al.).
- For parameter optimization, a tree is built by hierarchically splitting the parameter interval:
  - At an interior node, select a branch (i.e. subinterval) according to UCB, and descend.
  - At a leaf, split the leaf (i.e. split the interval) and sample from the 'unvisited' child node.

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

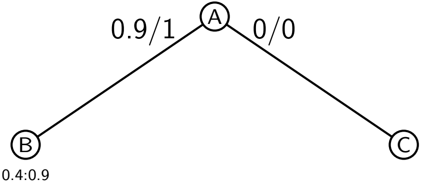
# Example for parameter optimization with UCT

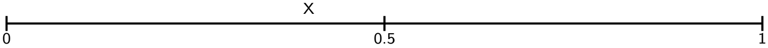


# Example for parameter optimization with UCT

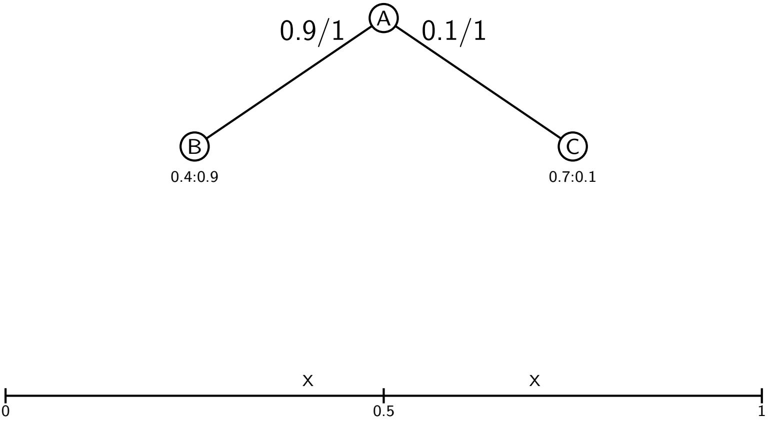


# Example for parameter optimization with UCT

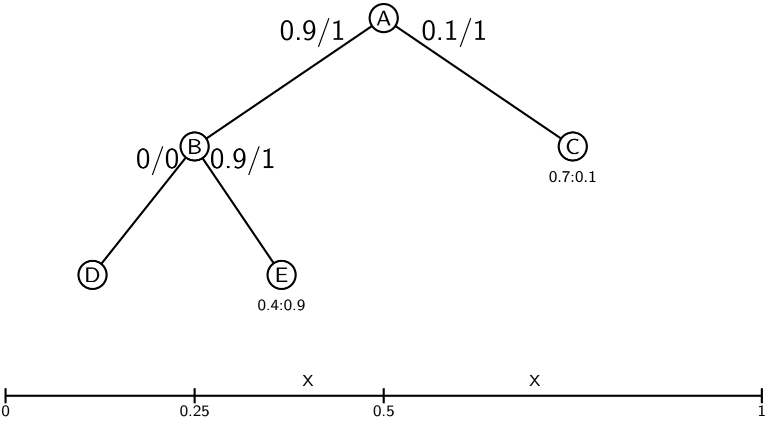


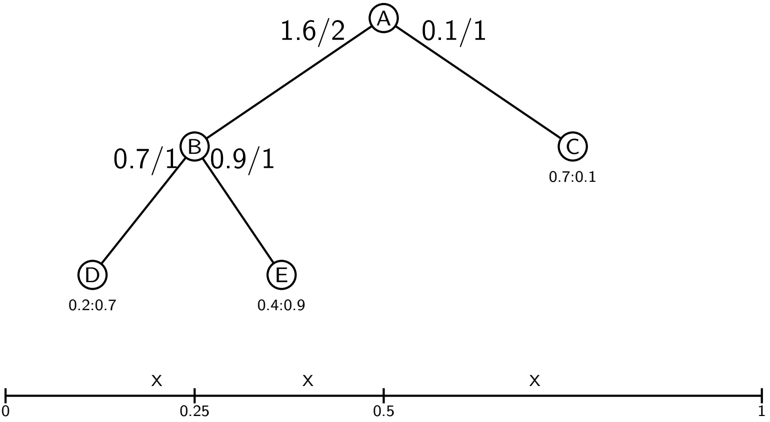
# Example for parameter optimization with UCT



# Example for parameter optimization with UCT



# Example for parameter optimization with UCT



# Application of UCT parameter optimization

- Churn prediction of a telecommunication company using the RPROP algorithm with 7 parameters.
- UCT converged to a good solution five times faster than RSPSA (Resilient Simultaneous Perturbation Stochastic Approximation).

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Application of UCT parameter optimization

- Churn prediction of a telecommunication company using the RPROP algorithm with 7 parameters.
- UCT converged to a good solution five times faster than RSPSA (Resilient Simultaneous Perturbation Stochastic Approximation).
- One RPROP run took approx. 12 hours. On 50 processors, the parameter tuning took approx 3 weeks.

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

# Apprenticeship learning using inverse reinforcement learning and gradient methods

Inverse reinforcement learning (IRL):

- How to imitate an observed (optimal) behavior of an expert?

# Apprenticeship learning using inverse reinforcement learning and gradient methods

Inverse reinforcement learning (IRL):

- How to imitate an observed (optimal) behavior of an expert?
- Imitate behavior in observed states!

# Apprenticeship learning using inverse reinforcement learning and gradient methods

Inverse reinforcement learning (IRL):

- How to imitate an observed (optimal) behavior of an expert?
- Imitate behavior in observed states!

- Problem: does not generalize well.

# Apprenticeship learning using inverse reinforcement learning and gradient methods

Inverse reinforcement learning (IRL):

- How to imitate an observed (optimal) behavior of an expert?
- Imitate behavior in observed states!

- Problem: does not generalize well.
- Extract a reward function that explains the observed behavior!

# Solving the IRL task

Consider (linearly) parameterized rewards, $r_\theta(s) = \sum_{i=1}^{n} \theta_i \phi_i(s)$.

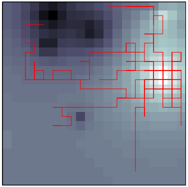
- Find a parameter vector $\theta$ which generates a behavior that is close to the observed expert behavior.
- Define closeness:

$$J(\theta) = \sum_{s \in \mathcal{S}} \mu_E(s)(\pi_\theta(s) - \pi_E(s))^2$$

  ($\mu_E$ – estimate of the expert's stationary distribution, $\pi_\theta$ – optimal policy for $\theta$, $\pi_E$ – expert's policy).
- Natural gradient techniques can be applied to improve performance.

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

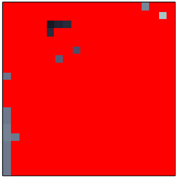
Expert trajectories



Low error region

Direct policy imitation

Low error region



IRL

Markov decision process (MDP) $M$:

$$\begin{aligned}
\mathcal{S} &\quad \ldots \quad \text{state space} \\
\mathcal{A} &\quad \ldots \quad \text{action space} \\
r(s,a) &\quad \ldots \quad \text{reward in } [0,1] \text{ for choosing} \\
&\qquad\qquad \text{action } a \text{ in state } s \\
p(s'|s,a) &\quad \ldots \quad \text{transition probability to state } s' \\
&\qquad\qquad \text{when choosing action } a \text{ in state } s' \\
\pi^* : \mathcal{S} \to \mathcal{A} &\quad \ldots \quad \text{optimal policy} \\
R_T(\pi) = \sum_{t=1}^{T} r(s_t, a_t) &\quad \ldots \quad \text{total reward of policy } \pi \text{ after } T \text{ steps,}
\end{aligned}$$

$s_t$ are the (random) states visited by $\pi$,

and $a_t$ are the chosen actions

# Undiscounted online regret

We are interested in the online regret of a strategy $\pi$:

$$\Delta_T(\pi) := R_T(\pi^*) - R_T(\pi)$$

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

# Undiscounted online regret

We are interested in the online regret of a strategy $\pi$:

$$\Delta_T(\pi) := R_T(\pi^*) - R_T(\pi)$$

Discounted regrets are not very useful for online analysis:
For $\gamma \in [0, 1)$,

$$\sum_{t=0}^{\infty} \gamma^t r(s_t^*, a_t^*) - \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) = O(1).$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Bounds on the regret

For $S = |\mathcal{S}|$ states and $A = |\mathcal{A}|$, our $\mathrm{UCRL}$ algorithm achieves

$$\Delta_T(\mathrm{UCRL}) = \tilde{O}\left(DS\sqrt{AT}\right),$$

where $D$ denotes the **diameter** of the MDP: This is the time, such that for any pairs of states $s_1, s_2 \in \mathcal{S}$ there is a policy which moves from $s_1$ to $s_2$ within $D$ steps on average:

$$D = \max_{s_1, s_2} \min_{\pi} \mathbb{E}\left[T(s_2|\pi, s_1)\right]$$

# Bounds on the regret

For $S = |\mathcal{S}|$ states and $A = |\mathcal{A}|$, our $\mathrm{UCRL}$ algorithm achieves

$$\Delta_T(\mathrm{UCRL}) = \tilde{O}\left(DS\sqrt{AT}\right),$$

where $D$ denotes the **diameter** of the MDP: This is the time, such that for any pairs of states $s_1, s_2 \in \mathcal{S}$ there is a policy which moves from $s_1$ to $s_2$ within $D$ steps on average:

$$D = \max_{s_1, s_2} \min_{\pi} \mathbb{E}\left[T(s_2|\pi, s_1)\right]$$

Matching lower bound:
There are MDPs such that for any algorithm

$$\Delta_T = \Omega\left(\sqrt{DSAT}\right).$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: PAC-like bounds

- $E^3$ by Kearns and Singh (1998):
  After $poly(1/\epsilon, S, A, T^{\epsilon}_{mix})$ steps the per-trial regret is at most $\epsilon$.
- Analysis of Rmax by Kakade (2003):
  Bound on the number of actions which are not $\epsilon$-optimal:

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(S^2 A (T^{\epsilon}_{mix}/\epsilon)^3\right)$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: PAC-like bounds

- $E^3$ by Kearns and Singh (1998):
  After $poly(1/\epsilon, S, A, T^{\epsilon}_{mix})$ steps the per-trial regret is at most $\epsilon$.
- Analysis of Rmax by Kakade (2003):
  Bound on the number of actions which are not $\epsilon$-optimal:

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(S^2 A (T^{\epsilon}_{mix}/\epsilon)^3\right)$$

- Our result gives

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(D^2 S^2 A/\epsilon^2\right).$$

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: PAC-like bounds

- $E^3$ by Kearns and Singh (1998):
  After $poly(1/\epsilon, S, A, T^{\epsilon}_{mix})$ steps the per-trial regret is at most $\epsilon$.
- Analysis of Rmax by Kakade (2003):
  Bound on the number of actions which are not $\epsilon$-optimal:

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(S^2 A (T^{\epsilon}_{mix}/\epsilon)^3\right)$$

- Our result gives

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(D^2 S^2 A/\epsilon^2\right).$$

- $T^{\epsilon}_{mix}$ is the number of steps such that for **any** policy $\pi$ its actual per-trial reward is $\epsilon$-close to the expected per-trial reward.

# Relation to other work: PAC-like bounds

- $E^3$ by Kearns and Singh (1998):
  After $poly(1/\epsilon, S, A, T^{\epsilon}_{mix})$ steps the per-trial regret is at most $\epsilon$.
- Analysis of Rmax by Kakade (2003):
  Bound on the number of actions which are not $\epsilon$-optimal:

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(S^2 A (T^{\epsilon}_{mix}/\epsilon)^3\right)$$

- Our result gives

$$\#\{t : a_t \neq a^{\epsilon}_t\} = \tilde{O}\left(D^2 S^2 A/\epsilon^2\right).$$

- $T^{\epsilon}_{mix}$ is the number of steps such that for **any** policy $\pi$ its actual per-trial reward is $\epsilon$-close to the expected per-trial reward.
- For small $\epsilon$, $T^{\epsilon}_{mix} > D/\epsilon$.

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: $\log T$ bounds

Assume that there is a **gap** $g$ between the average per-trial reward of the optimal and the 2nd best policy.

# Relation to other work: $\log T$ bounds

Assume that there is a **gap** $g$ between the average per-trial reward of the optimal and the 2nd best policy.

- We get

$$\mathbb{E}\left[\Delta_T(\mathrm{UCRL})\right] = O\left(\frac{D^2 S^2 A}{g^2} \log T\right).$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: log $T$ bounds

Assume that there is a **gap** $g$ between the average per-trial reward of the optimal and the 2nd best policy.

- We get

$$\mathbb{E}\left[\Delta_T(\mathrm{UCRL})\right] = O\left(\frac{D^2 S^2 A}{g^2}\log T\right).$$

- Burnetas, Katehakis, 1997, Tewari, Bartlett, NIPS 2007:

$$\mathbb{E}\left[\Delta_T\right] = O\left(\frac{D^2_{\max}|S||A|}{g^2}\log T\right)$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Relation to other work: $\log T$ bounds

Assume that there is a **gap** $g$ between the average per-trial reward of the optimal and the 2nd best policy.

- We get
$$\mathbb{E}\left[\Delta_T(\mathrm{UCRL})\right] = O\left(\frac{D^2 S^2 A}{g^2} \log T\right).$$
- Burnetas, Katehakis, 1997, Tewari, Bartlett, NIPS 2007:
$$\mathbb{E}\left[\Delta_T\right] = O\left(\frac{D_{\max}^2 |S||A|}{g^2} \log T\right)$$
- Main difference (recall $D = \max_{s_1,s_2} \min_\pi \mathbb{E}\left[T(s_2|\pi, s_1)\right]$):
$$D_{\max} = \max_{s_1,s_2} \max_\pi \mathbb{E}\left[T(s_2|\pi, s_1)\right]$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# The UCRL algorithm: Upper Confidence Reinforcement Learning

- The algorithm runs in rounds $k = 1, 2, \ldots$, each starting at some time $t_k$.
- A new round starts when the occurrences of some state-action pair $(s, a)$ have doubled,
$$N(s, a; t_{k+1}) = 2 \cdot N(s, a; t_k).$$
- Within a round, a fixed policy $\tilde{\pi}_k : \mathcal{S} \to \mathcal{A}$ is used.
- The policy $\tilde{\pi}_k$ is chosen such that it maximizes the expected reward for the best (maximal reward) *plausible* MDP, in respect to the current empirical estimates $\hat{p}(\cdot|s, a; t_k)$.
- An MDP $\tilde{M}$ is plausible if

$$||\tilde{p}(\cdot|s, a; t_k) - \hat{p}(\cdot|s, a; t_k)||_1 \leq \sqrt{\frac{const \cdot S}{N(s, a; t_k)}} \log t_k.$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: The bias $\lambda$

The policy $\tilde{\pi}_k$ can be calculated quite quickly by value iteration:

# Details of UCRL: The bias $\lambda$

The policy $\tilde{\pi}_k$ can be calculated quite quickly by value iteration:

- In discounted reinforcement learning we have Bellman updates

$$V(s) \leftarrow \max_a \left[ r(s,a) + \gamma \sum_{s'} V(s')p(s'|s,a) \right].$$

# Details of UCRL: The bias $\lambda$

The policy $\tilde{\pi}_k$ can be calculated quite quickly by value iteration:

- In discounted reinforcement learning we have Bellman updates

$$V(s) \leftarrow \max_a \left[ r(s,a) + \gamma \sum_{s'} V(s') p(s'|s,a) \right].$$

- In undiscounted reinforcement learning we can use *bias updates*

$$\lambda(s) \leftarrow \max_a \left[ r(s,a) + \sum_{s'} \lambda(s') p(s'|s,a) \right]$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: The bias $\lambda$

The policy $\tilde{\pi}_k$ can be calculated quite quickly by value iteration:

- In discounted reinforcement learning we have Bellman updates

$$V(s) \leftarrow \max_a \left[ r(s,a) + \gamma \sum_{s'} V(s')p(s'|s,a) \right].$$

- In undiscounted reinforcement learning we can use *bias updates*

$$\lambda(s) \leftarrow \max_a \left[ r(s,a) + \sum_{s'} \lambda(s')p(s'|s,a) \right]$$

and normalization

$$\begin{aligned} \rho &\leftarrow \min_{s'} \lambda(s') \\ \lambda(s) &\leftarrow \lambda(s) - \rho. \end{aligned}$$

# Details of UCRL: The bias $\lambda$

The policy $\tilde{\pi}_k$ can be calculated quite quickly by value iteration:

- In discounted reinforcement learning we have Bellman updates

$$V(s) \leftarrow \max_a \left[ r(s,a) + \gamma \sum_{s'} V(s')p(s'|s,a) \right].$$

- In undiscounted reinforcement learning we can use *bias updates*

$$\lambda(s) \leftarrow \max_a \left[ r(s,a) + \sum_{s'} \lambda(s')p(s'|s,a) \right]$$

and normalization

$$\begin{aligned} \rho &\leftarrow \min_{s'} \lambda(s') \\ \lambda(s) &\leftarrow \lambda(s) - \rho. \end{aligned}$$

- The bias update converges (for non-periodic MDPs) to $\lambda(\cdot)$ with $0 \leq \lambda(s) \leq D$.

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: Bias and regret

- The bias $\lambda(\cdot)$ solves the equation

$$\lambda(s) = \max_a \left[ r(s,a) - \rho^* + \sum_{s'} \lambda(s') p(s'|s,a) \right]$$

where $\rho^*$ is the optimal per-trial reward.

- The advantage of starting in state $s$ over starting in state $s'$ — followed by an infinite number of trials — is given by $\lambda(s) - \lambda(s')$.

# Details of UCRL: Bias and regret

- The bias $\lambda(\cdot)$ solves the equation

$$\lambda(s) = \max_a \left[ r(s,a) - \rho^* + \sum_{s'} \lambda(s') p(s'|s,a) \right]$$

where $\rho^*$ is the optimal per-trial reward.

- The advantage of starting in state $s$ over starting in state $s'$ — followed by an infinite number of trials — is given by $\lambda(s) - \lambda(s')$.
- For each time a non-optimal action $a \neq a^* = a^*(s)$ is chosen, a regret $\delta$ is suffered,

$$\begin{aligned} \delta &= r(s,a^*) - r(s,a) + \sum_{s'} \lambda(s')[p(s'|s,a^*) - p(s'|s,a)] \\ &\leq r(s,a^*) - r(s,a) + D\,||p(\cdot|s,a^*) - p(\cdot|s,a)||_1 \end{aligned}$$

# Details of UCRL: Analysis

We compare per-trial rewards $\tilde{\rho}_k$ and $\rho_k$ for the chosen policies $\tilde{\pi}_k$ in the optimistic MDP $\tilde{M}_k$ and in the true MDP, resp.

$$\mathbb{E}\left[\Delta_T\right] \approx \rho^* T - \sum_k \rho_k (t_{k+1} - t_k)$$

# Details of UCRL: Analysis

We compare per-trial rewards $\tilde{\rho}_k$ and $\rho_k$ for the chosen policies $\tilde{\pi}_k$ in the optimistic MDP $\tilde{M}_k$ and in the true MDP, resp.

$$\begin{aligned} \mathbb{E}[\Delta_T] &\approx \rho^* T - \sum_k \rho_k (t_{k+1} - t_k) \\ &\leq \sum_k (\tilde{\rho}_k - \rho_k)(t_{k+1} - t_k) \end{aligned}$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: Analysis

We compare per-trial rewards $\tilde{\rho}_k$ and $\rho_k$ for the chosen policies $\tilde{\pi}_k$ in the optimistic MDP $\tilde{M}_k$ and in the true MDP, resp.

$$
\begin{aligned}
\mathbb{E}[\Delta_T] &\approx \rho^* T - \sum_k \rho_k (t_{k+1} - t_k) \\
&\leq \sum_k (\tilde{\rho}_k - \rho_k)(t_{k+1} - t_k) \\
&\leq \sum_k \sum_{t=t_k}^{t_{k+1}-1} D \, ||\tilde{p}(\cdot|s_t, a_t) - p(\cdot|s_t, a_t)||_1
\end{aligned}
$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: Analysis

We compare per-trial rewards $\tilde{\rho}_k$ and $\rho_k$ for the chosen policies $\tilde{\pi}_k$ in the optimistic MDP $\tilde{M}_k$ and in the true MDP, resp.

$$\begin{aligned}
\mathbb{E}\left[\Delta_T\right] &\approx \rho^* T - \sum_k \rho_k (t_{k+1} - t_k) \\
&\leq \sum_k (\tilde{\rho}_k - \rho_k)(t_{k+1} - t_k) \\
&\leq \sum_k \sum_{t=t_k}^{t_{k+1}-1} D \,||\tilde{p}(\cdot|s_t, a_t) - p(\cdot|s_t, a_t)||_1 \\
&= \sum_k \sum_{t=t_k}^{t_{k+1}-1} \tilde{O}\left( D \sqrt{\frac{S}{N(s_t, a_t; t_k)}} \right)
\end{aligned}$$

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

# Details of UCRL: Analysis

We compare per-trial rewards $\tilde{\rho}_k$ and $\rho_k$ for the chosen policies $\tilde{\pi}_k$ in the optimistic MDP $\tilde{M}_k$ and in the true MDP, resp.

$$
\begin{aligned}
\mathbb{E}[\Delta_T] &\approx \rho^* T - \sum_k \rho_k (t_{k+1} - t_k) \\
&\leq \sum_k (\tilde{\rho}_k - \rho_k)(t_{k+1} - t_k) \\
&\leq \sum_k \sum_{t=t_k}^{t_{k+1}-1} D \, ||\tilde{p}(\cdot|s_t, a_t) - p(\cdot|s_t, a_t)||_1 \\
&= \sum_k \sum_{t=t_k}^{t_{k+1}-1} \tilde{O}\left( D \sqrt{\frac{S}{N(s_t, a_t; t_k)}} \right) \\
&= \sum_{s,a} \tilde{O}\left( D \sqrt{S \cdot N(s, a; T)} \right) = \tilde{O}\left( DS\sqrt{AT} \right)
\end{aligned}
$$

PASCAL
Pattern Analysis, Statistical Modelling and Computational Learning

# Future research

# Future research

- **Tracking changes:**
  Allow changes in the MDP which need to be picked up by the learning algorithm.

# Future research

- **Tracking changes:**
  Allow changes in the MDP which need to be picked up by the learning algorithm.

- **Continuous state/action spaces:**
  - Difficult in a general setting
  - Progress for simplified (bandit) setting
  - Extend this to more interesting settings

# Future research

- **Tracking changes:**
  Allow changes in the MDP which need to be picked up by the learning algorithm.
- **Continuous state/action spaces:**
  - Difficult in a general setting
  - Progress for simplified (bandit) setting
  - Extend this to more interesting settings
- **Autonomous rewards:**
  Design autonomous reward functions which drive both the consolidation and the extension of learned knowledge, mimicking cognitive behavior.