Seminar: Reinforcement Learning in Information Retrieval

Dorota Glowacka (glowacka@cs.helsinki.fi)

Joel Pyykko (jgpyykko@cs.helsinki.fi)

Organisational Details and Assessment

- The seminar is worth 3 course points
- Duration: periods 3 and 4 (18.01 - 03.05)
- Aim of the seminar: introduction of the basic concepts of RL and fields where RL techniques are commonly used with an emphasis on information retrieval
- Assessment: a 8-10 page report on a topic related to RL in IR and one presentation on the subject of the report
- The topics can be chosen freely but there is also a list of suggested are ready topics

Schedule

Possible changes to schedule will appear on the course pages.

18.1. - 29.1.	Introductory lectures
30.1.	Deadline for topic selection
15.2.	Presentation of the chosen topic, 5 minutes, ~5 slides.
29.3.	Feedback session.
5.4.	Feedback session.
12.4.	Final presentations, part 1. 20 minutes, ~20 slides.
19.4.	Final presentations, part 2. 20 minutes, ~20 slides.
26.4.	Final presentations, part 3. 20 minutes, ~20 slides.
3.5	Deadline for the final paper submission.

Reinforcement Learning

Definition (Reinforcement Learning [Sutton and Barto, 1998])

“Reinforcement learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.”

Reinforcement learning - Introduction

How can an agent learn to choose optimal actions in each state to achieve its goals?

→ Learning from interaction

→ Reward and punishment

- RL agent learns by interacting with the environment and observing the consequences of its actions (reward or punishment).

- The agent must be able to sense the state of the environment to some extent and must be able to take actions that affect the state

- The agent must have a goal or goals relating to the state of the environment

- The agent must be able to learn from its own experience.

Examples 1

- A master chess player makes a move. The choice is informed both by planning--anticipating possible replies and counterreplies--and by immediate, intuitive judgments of the desirability of particular positions and moves.
- An adaptive controller adjusts parameters of a petroleum refinery's operation in real time. The controller optimizes the yield/cost/quality trade-off on the basis of specified marginal costs without sticking strictly to the set points originally suggested by engineers.

Examples 2

- A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour.
- A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past.

Examples 3

Walking robot

http://www.youtube.com/watch?v=iNL5-0_T1D0

Walking robot dog

http://www.youtube.com/watch?v=I4qQXP8FbnI

Robot learns to flip pancakes

http://www.youtube.com/watch?v=W_gxLKSsSIE

Elements of Reinforcement Learning

- A *policy* defines the learning agent's way of behaving at a given time.
- A *reward function* defines the goal in a reinforcement learning problem.
- A *value function* specifies what is good in the long run. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the *long-term* desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.
- A *model* of the environment mimics the behavior of the environment. Models are used for *planning*, i.e. deciding on a course of action by considering possible future situations before they are actually experienced.

Applications

- Robotics
- Games
- Auctions and pricing
- Information retrieval
- Industrial/automotive control
- Autonomous vehicles control
- Logistics
- Telecommunication networks
- Sensor networks
- Ambient intelligence
- Finance

Reinforcement Learning in Information Retrieval

- User modelling and result personalisation
- Exploratory search
- Online document ranking
- Multimedia retrieval: images, video, music
- Recommender systems
- Evaluation of ranking algorithms

Exploration vs Exploitation

The bandit problem

- You have a choice among *n* different options or actions.

- After each choice you receive a numerical reward chosen from a stationary probability distribution.

- Your objective is to maximize the expected total reward over some time period, e.g. over 1000 action selections (plays).

- Each action has an expected or mean reward (value of that action) which is unknown to the player.

Exploration vs Exploitation

- If you maintain estimates of the action values and always select the action whose estimated value is the greatest, then you select a greedy action, or you are *exploiting* your current knowledge of the values of the actions.

- If instead you select one of the nongreedy actions, then you are *exploring* because this enables you to improve your estimate of the nongreedy action's value.

- Exploitation is the right thing to do to maximize the expected reward on the one play, but exploration may produce the greater total reward in the long run.

Balancing Exploration and Exploitation

- Greedy methods
- Softmax action selection
- Linear reward penalty (learning automata methods)
- Incremental computation of action values
- Initial value estimates
- Reinforcement comparison methods
- Pursuit methods
- Associative search

Reading

Sutton & Barto “Reinforcement Learning: An Introduction” chapters 1 and 2

Recommended: exercises in section 2

Seminar: Reinforcement Learning and Applications

Dorota Glowacka (glowacka@cs.helsinki.fi)

Joel Pyykko (jgpyykko@cs.helsinki.fi)

Round 2

Reinforcement learning: Definitions

Reinforcement learning is a simple framing of the problem of *learning from interaction.*

It is automated decision making in a setting where we don’t yet know how everything works.

As it is unknown, *finding out* where the goodies are is important.

Reinforcement learning:
Some Basic Terminology

- The learner or the decision maker is called the ***agent***.

- The agent interacts with the ***environment***, comprising everything outside the agent.

- The agent and the environment interact continually by the agent selecting ***actions*** and the environment responding to these actions as ***states***.

- The environment also gives ***rewards*** to the agent, thus directing learning.

- The agent tries to find a ***policy*** of moves that maximizes the rewards on a long run.

- Figuring out how much the agent needs to know more about the world is known as ***exploration/exploitation -dilemma.***

The environment

Is often portrayable as one or the other:

- Markov Decision Process (MDP)
- Multi-armed Bandits

Markov Decision Process

- State space **S**, the various states the agent may be in.
- Actions **A**, the set of actions available to the agent
- Stochastic transition probabilities **T(s,a,s')** from a state-action pair to some other state.
- A reward function which we are trying to estimate is **R(s,a,s')** (note, expected reward associated with a state-action pair is usually **R(s,a)**).
- **R(s,a,s')** and **T(s,a,s')** are initially unknown (If they were not, we wouldn't need RL).
- Objective to find sequences of actions that give the most rewards.

Reinforcement learning in MDP - State value (policy)

A policy **π** is the mapping of actions to states, telling the agent what to do in each situation:

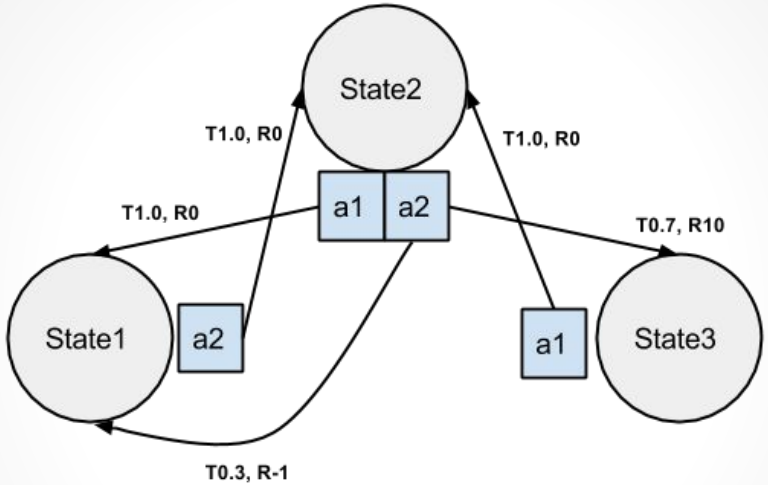
$$\pi(s) = \arg\max_a \Sigma_{s'} T(s,a,s') [R(s,a,s') + \gamma V_\pi(s')]$$

- $V_\pi(s')$ is the value of a consecutive state.

- Discount factor $0 \leq \gamma \leq 1$, gives less value to further away rewards.

The objective is to find the optimal policy by exploring the dynamics of the state space.

Reinforcement learning in MDP - An example



-An MDP with two actions, a1 = walk back, a2 = walk forward. From state2 when moving forward a 30% to get lost and move back to state1 instead.
-Rewards for getting to state3 is 10, returning to state1 is -1, otherwise 0.
-Optimal policy: S1->a2, S2->a2, S3->a1.

MDP Learning - State-action value (Q-learning)

The main equation is:

$Q(s,a) = R(s,a) + \gamma \Sigma_{s'} T(s,a,s') Q(s',a')$

- Often implemented as a matrix of **S x A**, each cell representing the expected Q-value of taking action **a** in state **s**.

- Value of a cell is updated as the state-action pair is perceived.

- Discount factor **$0 \leq \gamma \leq 1$**, gives less value to further away rewards.

- Optimal behavior, or policy, **$a = \text{argmax}_a Q(s,a)$**.

Q-learning - Example

Flappy:

https://www.youtube.com/watch?v=79BWQUN_Njc

MarioIO

https://www.youtube.com/watch?v=qv6UVOQ0F44

Multi-armed Bandits

- No state space, rather the agent is always in a single state.
- Actions **K**, the set of actions (slot-machines) available to the agent.
- Reward probability distributions $\mathbf{P_K(p_1, p_2,... p_k)}$ for each arm, initially unknown. Usually each arm has a probability of giving either 0 or 1 reward.
- Objective is to maximize rewards over a number of **H** pulls (a horizon).
- Simplified environment for exploration/exploitation dilemma: when to explore for better rewards, and when to exploit what we know already.

Bandits - Regret

Regret is $\rho = T\mu^* - \sum^{t}_{t=1} Tr^t$, where μ^* is the maximal reward mean, **T** is the rounds we've done thus far.

So basically: what if we had done the optimal move minus what we actually did.

With this it is easy to see how much we would lose rewards if we would go for exploration instead.

UCB - Algorithm

Upper Confidence Bounds, a baseline algorithm for “solving” bandits problems.

Deterministic policy: UCB1.
Initialization: Play each machine once.
Loop:

- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine j, n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

http://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/

Bandits - UCB

This defines an upper bound for regret. Warning, ‘graphic content’.

Theorem 1. *For all $K > 1$, if policy* UCB1 *is run on K machines having arbitrary reward distributions $P_1, \ldots, P_K$ with support in $[0, 1]$, then its expected regret after any number n of plays is at most*

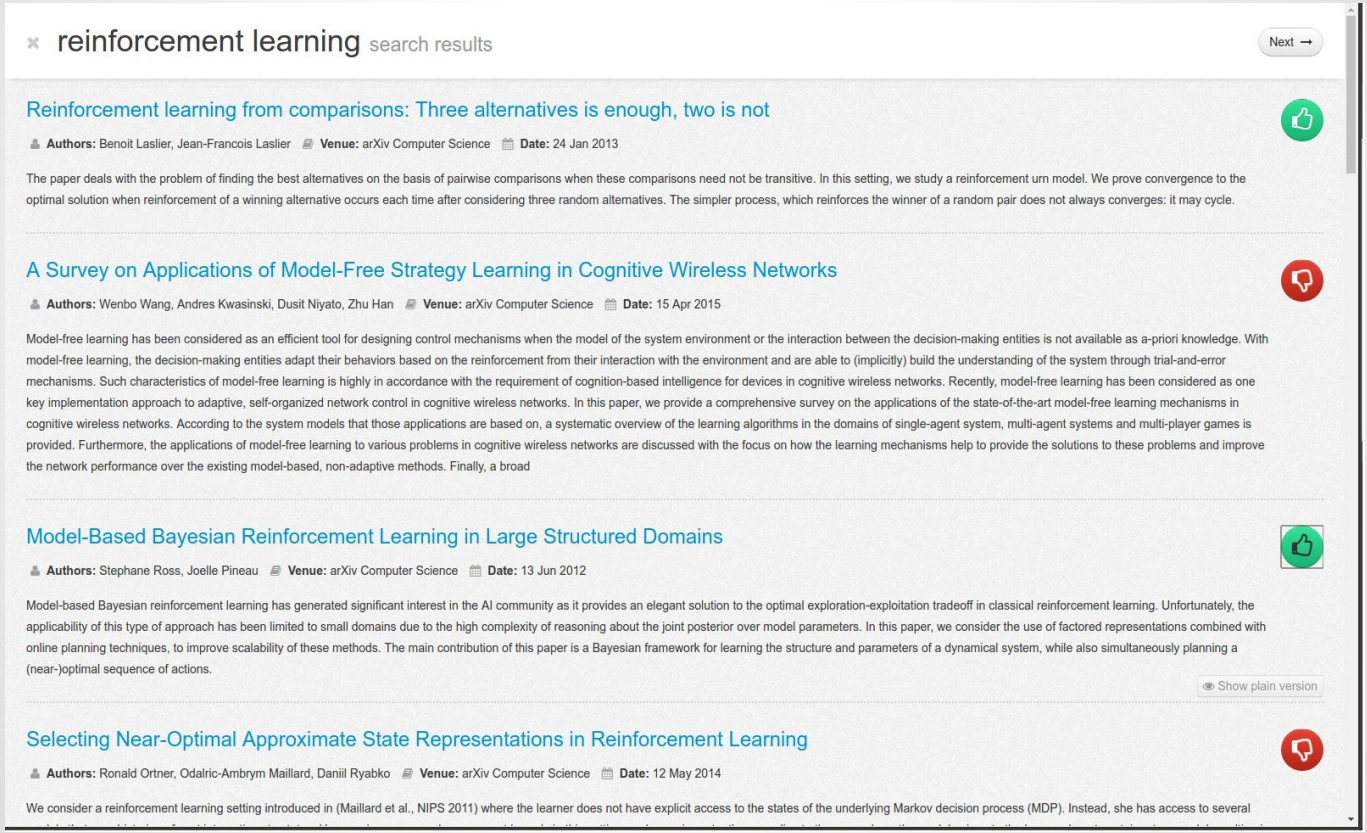
$$\left[8 \sum_{i:\mu_i<\mu^*} \left(\frac{\ln n}{\Delta_i}\right)\right] + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{j=1}^{K} \Delta_j\right)$$

where $\mu_1, \ldots, \mu_K$ are the expected values of $P_1, \ldots, P_K$.

Basically, it is logarithmic in time, without knowledge of ***P***.

Theoretically it is tighter than nothing, but it is still pretty loose.

Bandits - An Example



Other interesting topics

- Model-based reinforcement learning
  - Decision trees
  - Monte Carlo -methods
- Multi-agent systems
- Partial information learning
- With deep learning
  - Generative models
  - Value function learning
- Epsilon greedy exploration

Scheduling and Topics

- Chance to agree on presentation times

- Help and discussion on chosen topics

https://docs.google.com/spreadsheets/d/1awyodek3wEXOIlv-gD3wfzPK91Fd3yK9ljod033o_TI/edit#gid=0

Model-based reinforcement learning

- Learn a model of the transitions **T** and reward function **R**.
- Simulates the environment on what could happen.
- The **Q**-values are not saved into a matrix, so the agent must calculate them.
- Planning now possible in stochastic environments.
- Similarities with Monte Carlo planning.