MVA, Reinforcement Learning	**TP1**
Dynamic Programming and Reinforcement Learning	
Lecturer: *Emilie Kaufmann*	*emilie.kaufmann@inria.fr*

Report Deadline: November 13, midnight

Note: Your report should be *short* and based on the answers to questions Q1-Q5

Report and code should be sent by e-mail to emilie.kaufmann@inria.fr (one pdf file *yourname_TP1.pdf* + one archive *yourname_TP1.zip*), with [MVA 2016] in the title.

1 The One-Site Tree Cutting Problem

We would like to formalize the *tree cutting* problem and compute the strategy which maximizes the revenue. A tree keeps growing over time with a rate which may depend on the weather and it stops when it reaches a certain maximum height. At the same time the tree may get a disease, in which case it dies and looses all its value. When the company decides to cut a tree, it gains an amount of money which is proportional to the height of the tree. Whenever a tree is cut (or it is dead), a new tree has to be planted with a fixed cost. Knowing that the one unit of money looses value over time, find the optimal cutting strategy.

1.1 A Bit More Formal Definition of the Environment

- State space: the (discrete!) height of the tree (constrained to a maximum height)
- Initial state: the height of the tree is set to one
- Action space either cut or not the tree
- Dynamics:
  - If no cut: the tree grows up to a maximum height by a number of units which depend on the (random!) weather. It may also (randomly!) get a disease.
  - If cut: a new tree is planted with an initial height of one unit.
- Reward:
  - If no cut: a fixed amount of maintenance cost
  - If cut: the value of each unit of wood times the height of the tree minus the cost of planting a new tree.
- Discount factor: we assume a bank interest rate $r = 0.05$, and so discount factor is set of $\gamma = 1/(1+r)$.

1.2 Work to do

1. Formalize the problem more precisely (some decisions are of course arbitrary, such as the influence of the weather on the growth) and implement two functions:

   (a) tree_sim which receives as input a state and an action and it returns the next state and the reward.

   (b) tree_MDP which returns the dynamics and the reward function (in suitable structures).

   Q1: Explicit the MDP and the parameter chosen to model the random effects.

2. Policy evaluation: define an arbitrary policy and evaluate it in the initial state using one RL method (Monte-Carlo or TD(0)) and one dynamic programming method (matrix inversion or Bellman operator).

   - Q2: If V_n denotes the value function computed by the RL method based on n trajectories, chart $V_n(x_0) - V^\pi(x_0)$, where $x_0 = I$ and V^π is the value function computed with DP.

   Notice: for Monte-Carlo at each repetition you will obtain different estimates since the transitions are random, so multiple runs of the same experiments are needed. Be sure to plot the average result.

3. Optimal policy:

   - Q3: Compute the optimal policy with the two dynamic programming method seen in class, Policy Iteration and Value Iteration.

     Recall that both VI and PI can be implemented using the Q-value function associated to a value function V, defined by

     $$Q(x,a) = r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y|x,a) V(y).$$

   - Q4: For both methods, plot $||V^* - V_k||_\infty$ as a function of iteration k to compare the speed of convergence and discuss the relative merits of the two approaches.

     For Policy Iteration, $V_k = V^{\pi_k}$, where π_k is the policy obtained after k iterations.

   - Q5: Compute the optimal policy with Q-learning, a reinforcement learning algorithm. Comment on the performance of Q-Learning using at least the first metric below.

Notes on the implementation of Q-learning

- Start from I
- Select an action as $a^+ = \arg\max Q(s,a)$ with probability $1-\epsilon$ and randomise with probability ϵ

Q-Learning works in episodes (indexed by n) starting from the initial state up to a maximal horizon. The cumulated discounted reward obtained in each episode can also be measured.

Measure of performance for Q-learning, at the end of each episode:

- Performance in the initial state $|V^*(I) - V^{\pi_n}(I)|$, where π_n is the greedy policy w.r.t. Q_n
- Performance over all the other state $||V^* - V^{\pi_n}||_\infty$
- Reward cumulated over the episode

2 Going further

1. Study how the obtained results change when changing some of the parameters of the problem (initial height, cost of planting a new tree, gain in selling a tree, and so on).

2. Consider the case where we have two sites where we can grow trees. At each point in time, the decision is whether to cut a tree and which one and the state should consider both sites. Implement the extension or discuss how it could be implemented.

3. Propose a model (and test Q-learning on it) to solve the problem sketched here `http://stackoverflow.com/questions/8337417/markov-decision-process-value-iteration-how-does-it-work`