# 17044633_RL_hw1

February 20, 2018

By: Yuan Zhang
SN: 17044633

## 1 RL homework 1

**Due date: 19 February 2017, 11:55pm (just before mid-night!)**

### 1.1 Assignment 4a:

In the next text field, write down the update function to the preferences for all actions $\{a_1, \ldots, a_n\}$ if you selected a specific action $A_t = a_i$ and received a reward of $R_t$.

In other words, complete:

$$\begin{aligned} p_{t+1}(a) &= \ldots && \text{for } a = A_t \\ p_{t+1}(b) &= \ldots && \text{for all } b \neq A_t \end{aligned}$$

**[10 pts] Instructions**: please provide answer in markdown below.

$$\begin{aligned} p_{t+1}(a) &= p_t(a) + \alpha R_t(1 - \pi_t(a)) && \text{for } a = A_t \\ p_{t+1}(b) &= p_t(b) - \alpha R_t \pi_t(b) && \text{for all } b \neq A_t \end{aligned}$$

## 2 Assignment 5: Analyse Results

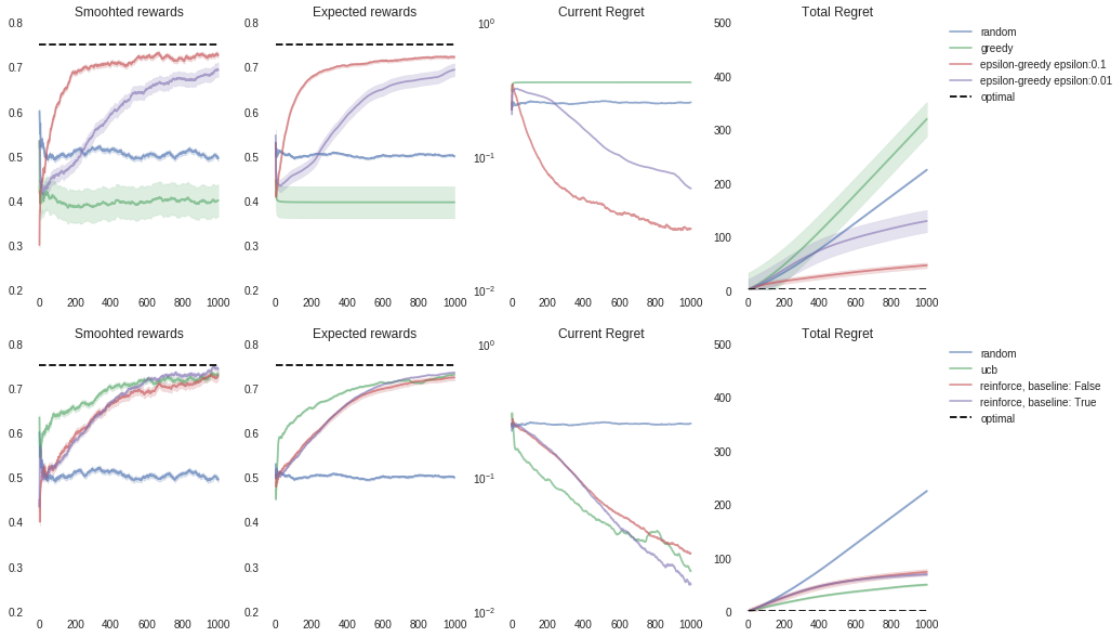
#### 2.0.1 Run the cell below to train the agents and generate the plots for the first experiment.

Trains the agents on a Bernoulli bandit problem with 5 arms, with a reward on success of 1, and a reward on failure of 0.

In [35]:
```
#@title Experiment 1: Bernoulli bandit

number_of_arms = 5
number_of_steps = 1000

agents = [
    Random(number_of_arms),
```

```
        Greedy(number_of_arms),
        EpsilonGreedy(number_of_arms, 0.1),
        EpsilonGreedy(number_of_arms, 0.01),
        UCB(number_of_arms),
        REINFORCE(number_of_arms),
        REINFORCE(number_of_arms, baseline=True),
    ]

    train_agents(agents, number_of_arms, number_of_steps)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:32: MatplotlibDeprecationWarning:
```




## 2.1 Assignment 5 a.

(Answer inline in the markdown below each question.)

**[5pts]** Name the best and worst algorithms, and explain (with one or two sentences each) why these are best and worst.

*The worst algorithm is greedy and the best is UCB. We can see this from the total regret where greedy is the highest and UCB is the lowest. Greedy doesn't consider exploration, which makes it hardly choose optimal action. UCB balance exploraion and exploitation.*

**[5pts]** Which algorithms are guaranteed to have linear total regret?

*Random, greedy and epsilon-greedy have linear total regret.*

**[5pts]** Which algorithms are guaranteed to have logarithmic total regret?

*UCB and reinfore have logarithmic total regret.*

**[5pts]** Which of the $\epsilon$-greedy algorithms performs best? Which should perform best in the long run?

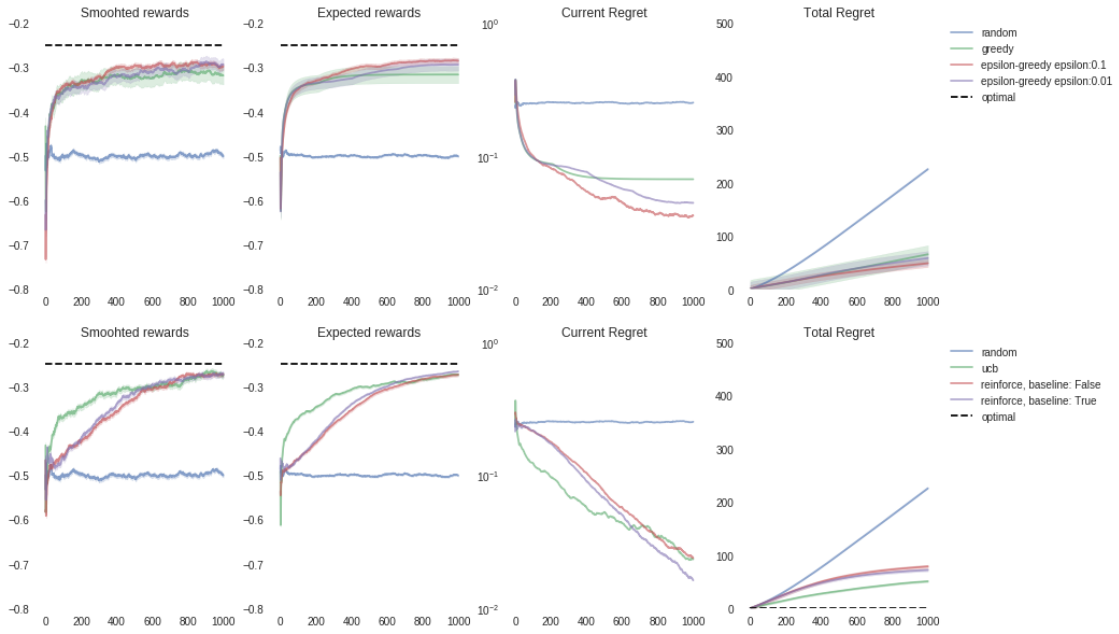
*epsilon=0.1 performs better. For the long run, epsilon=0.01 is better. Because at the beginning, we'd better explore more actions. After some time, we already have enough knowledge and we'd better be greedy.*

### 2.1.1 Run the cell below to train the agents and generate the plots for the second experiment.

Trains the agents on a bernoulli bandit problem with 5 arms, with a reward on success of 0, and a reward on failure of -1.

```
In [38]: #@title Experiment 2: R = 0 on success, R = -1 on failure.
         number_of_arms = 5
         number_of_steps = 1000

         train_agents(agents, number_of_arms, number_of_steps,
                      success_reward=0., fail_reward=-1.)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:32: MatplotlibDeprecationWarning:
```




## 2.2 Assignment 5 b.

(Answer inline in the markdown.)

**[10pts]** Explain which algorithms improved from the changed rewards, and why.

(Use at most two sentences per algorithm and feel free to combine explanations for different algorithms where possible).

*Greedy and epsilon-greedy improve in this setting. Because for this setting, average of the sampled rewards are all less than(or equal to) 0. So this makes greedy alogithm to choose actions that haven't chose before, which is a kind of exploration. For the epsilon-greedy, the reason is similar. Negetive sampled rewards make alogithm to explore new actions.*