Decision Making 1

Noel Welsh

04 November 2010

Announcements

- Take broken kit directly to Bert for faster turn around.

Recap

- Last lecture we saw that the behavioural robotics provides a good framework for constructing low level and intermediate level behaviours for a mobile robot.
- However, the methodology of behavioural robotics is lacking. If were are to create truly intelligent robots they must be able to learn behaviours. This is what we (start to) look at today.

The Problem Setup

- Today we're going to consider a relatively simple task, that of choosing between a fixed number of pre-defined behaviours. This a reasonable thing to do as we've seen we can relatively easily create low-level behaviours.
- Example: Insect robot
- There are two important restrictions that we will use: we assume that the action choice makes no use of observations, and that actions do not significantly affect the world.

The Multi-armed Bandit

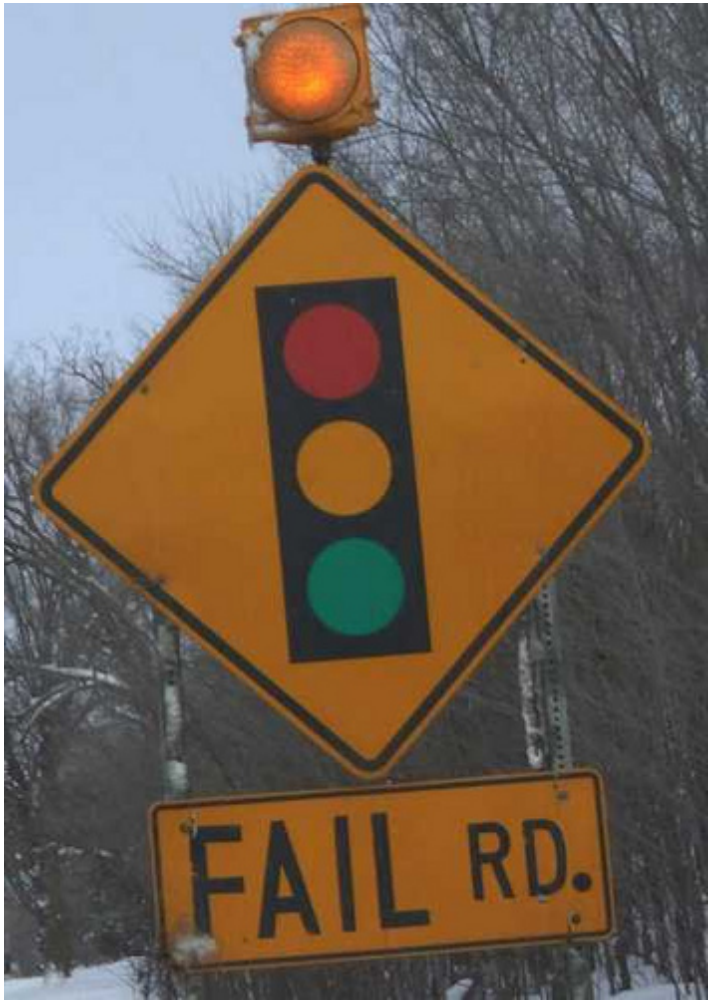
- This known as the *multi-armed bandit*.
- The names comes from the “one-armed bandit” also known as fruit, slot, or poker machines.
- The multi-armed bandit problem has important applications outside of robotics. For example, choosing the best advertisement to display to a web user is a multi-armed bandit problem. This task was worth approximately 23 *billion* dollars to Google in 2009. Choosing between different drugs to treat a patient is also a multi-armed bandit problem.

Getting Formal

- We need to formalise our measure of success a bit before we can examine different algorithms for the bandit problem.
- We want to create an algorithm that wins as much as possible.

Win

Fail



No Regrets

- Each time we choose an action we get a *reward*, which for convenience we say is a number between 1 and 0. The higher the reward the higher our win. Rather than maximising reward it is easier to minimise our *regret*, the amount we fail compared the best possible action.
- We denote the average (or mean or expected) reward of the best action as μ^* and of any other action j as μ_j. There are a total of K actions. We write $T_j(n)$ for the number of times we have tried action j in a total of n action. Formally, the regret after n actions is defined as

$$regret(n) = \mu^* n - \sum_{j=1}^{K} \mathbb{E}[T_j(n)] \tag{1}$$

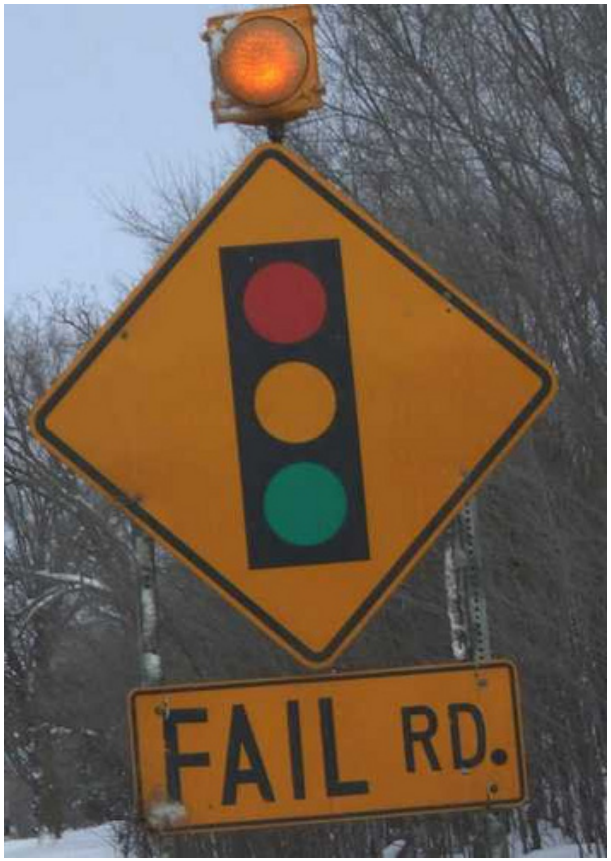
Your Turn

Try to come up with an algorithm for the multi-armed bandit problem. You might find it helpful to think about hypothesis testing and the Central Limit Theorem. How does your algorithm behave with respect to regret?

Hypothesis Testing

- One approach to the bandit problem is to frame it as a hypothesis testing problem.

Oh Dear



Hypothesis Testing Fail

- We make no use of the information we gain during the trial. Thus our regret will be higher than it need be.
- We may not be able to see a statistically significant result. Then what do we do?
- Most hypothesis tests are setup to minimise Type 1 errors (falsely rejecting H_0) at the expense of increasing Type 2 errors (falsely accepting H_0). This doesn't seem a sensible thing to do in this case. We can adjust the probability of the different errors by changing the p-value, but as we decrease one probability we increase the other. The essential problem here is we are forcing ourselves to make a hard cut-off when really we can keep trying different actions indefinitely.
- (Doesn't stop people doing it.)

Exploring and Exploiting

- An alternative we might try is to choose the action with the highest average reward so far. This seems like it would work but can give regret the scales linearly with the number of plays.
- This illustrates a classic problem which is the defining characteristic of *decision making*: the trade-off between exploring and exploiting. Exploring means to try new actions to learn their effects. Exploiting means to try what we know how worked in the past.

ϵ-greedy

- A simple modification to the above algorithm is to choose the best action with a probability $1 - \epsilon$, and to otherwise choose an action at random. This is known as ϵ-greedy.
- If we keep ϵ constant the regret will grow linearly with the number of actions.
- A better way is to use some function $\epsilon(t)$ that decreases over time. This makes our algorithm shift from exploration to exploitation as our estimates of the rewards improve.

Optimisim FTW

- The ϵ-greedy algorithm could still be improved. As given it explores uniformly over all actions. Better would be to use our estimates of reward to guide exploration.
- One way to do this is to try each action with a probability proportional to our current estimate of reward. However we can do better than this!
- If we have tried an action less often then our estimated reward is less accurate. One approach is to add on a certain amount to each reward, giving an upper bound on where we expect the true average reward to lie. The amount we add should decrease with the number of times we have tried an action. This is called an *optimistic* policy.

Chernoff-Hoeffding Bound

- How do we know how much to add? We can turn to the classic Chernoff-Hoeffding bound to get (most of the way to) an answer.
- Let $X_1, X_2, \ldots, X_n$ be independent random variables in the range $[0, 1]$ with $\mathbb{E}[X_i] = \mu$. Then for $a > 0$,

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq \mu + a\right) \leq e^{-2a^2 n} \tag{2}$$

- The other side also holds:

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \leq \mu - a\right) \leq e^{-2a^2 n} \tag{3}$$

- If we wanted to put an upper bound of p on the average reward, we can solve $p = e^{-2a^2 n}$ for a to find out how much we should add. Try this.

UCB1 Overview

- The algorithm UCB1 (for *upper confidence bound*) is an algorithm for the multi-armed bandit that achieves regret the grows only logarithmically with the number of actions made.
- It is also dead-simple to implement, so good for small robots.

UCB1 Details

- For each action j record the average reward $\bar{x}_j$ and number of times we have tried it n_j. We write n for total number of actions we have tried.
- Try the action that maximises $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$
- That is all! What is the confidence bound we're using?

UCB1's Explore/Exploit Tradeoff

- From out analysis of the Chernoff-Hoeffing bound above we can see that the confidence bound grows with the total number of actions we have taken but shrinks with the number of times we have tried this particular action. This ensures each action is tried infinitely often but still balances exploration and exploitation.

UCB1 Regret Bound

- The regret for UCB1 grows at a rate of ln n. In particular, after n actions it is at most

$$\sum_{j=1}^{K} \frac{4 ln n}{\Delta_j} + \left(1 + \frac{\pi^2}{3}\right) \Delta_j \tag{4}$$

where $\Delta_j = \mu^* - \mu_j$.

Extensions

- We can consider observations of the world when making a decision. This is known as the *contextual bandit*.
- There are many variants of the bandit algorithm that address, e.g., costs for switching between arms, or arms with finite lifespan.
- An important variant is the *non-stochastic bandit* which makes no assumptions about the reward distribution (not even identically distributed). This is the Exp3 family of algorithms.
- UCB1 is the building block for tree search algorithms (e.g. UCT) used to, e.g., play games
- Considering the effect of sequence of decisions (i.e. allowing decisions to effect the world) is *reinforcement learning*.

Notes

- These notes also cover most of the same material as us:

http://www.cs.caltech.edu/courses/cs253/slides/cs253-07-ucb1.pdf

- Fail Road by Firefly the Great (Dagny Scott)

http://www.flickr.com/photos/fireflythegreat/2845637227/

- Lynx High Five by Amadeus Varadi Hellequin

http://www.flickr.com/photos/rucken/4411674785