Robotics
Exercise 13

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

January 24, 2014

On Wed. 29th we meet as usual for the exercise.
On Tue. 28th, 14:00 my office, interested students are invited to try whatever they like on the racer hardware, play around, etc.

1 Policy Search for the Racer

We consider again the simulation of the racer as given in `09-racer` in your code repo.
In this exercise the goal is to find a policy

$$\pi:\ \phi(y) \mapsto u = \phi(u)^\top w$$

that maps some features of the (history of the) direct sensor signals y to the control policy.
Use black-box optimization to find parameters w that robustly balance the racer.
Some notes:

- Features: In the lecture I suggested that a range of interesting features is:

  $$\Big(y_t, \dot{y}_t, \langle y\rangle_{0.5}, \langle \dot{y}\rangle_{0.5}, \langle y\rangle_{0.9}, \langle \dot{y}\rangle_{0.9}, u_t, u_{t-1}\Big)$$

  However, I noticed that a balancing policy can also be found for the direct sensor signals only, that is:

  $$\phi(y) = (1, y) \in \mathbb{R}^5$$

  (the augmentation by 1 is definitly necessary).

- Cost function: Realistically, the running costs c_t would have to be defined on the sensor signals only. (On the real robot we don't know the real state – if the robot is to reward itself it needs to rely on sensor signals only.) I tried

  ```
  costs += .1*MT::sqr(y(3)) + 1.*MT::sqr(y(2));
  ```

  which combines the wheel encoder `y(3)` and the gyroscope reading `y(2)`. That worked mediocre. For a start, cheat and directly use the state of the simulator to compute costs:

  ```
  costs += 1.*MT::sqr(R.q(0)) + 10.*MT::sqr(R.q(1));
  ```

- Episodes and duration costs: To compute the cost for a given w you need to simulate the racer for a couple of time steps. For the optimization it is really bad if an episode is so long that it includes a complete failure and wrapping around of the inverted pendulum. Therefore, abort an episode if `fabs(R.q(1))` too large and penalize an abortion with an extra cost, e.g., proportional to $T - t$. Try different episode horizons T up to 500; maybe increase this horizon stage-wise.

- Optimizer: You are free to use any optimizer you like. On the webpage you find a reference implementation of CMA by Niko Hansen (with wrapper using our `arr`), which you may use. In that case, add `cmaes.o` and `search_CMA.o` in the `Makefile`. The typical loop for optimization is

```
SearchCMA cma;
cma.init(5, -1, -1, -0.1, 0.1);
arr samples, values;

uint T=500;
for(uint t=0;t<1000;t++){
  cma.step(samples, values);
  for(uint i=0;i<samples.d0;i++) values(i) = evaluateControlParameters(samples[i], T);
  uint i=values.minIndex();
  cout <<t <<' ' <<values(i) <<' ' <<samples[i] <<endl;
}
```