How can we design a more intelligent path-planning?

Akira Imada

Brest State Technical University
Moskowskaja 267, Brest 224017 Republic of Belarus
akira@bsty.by

In the "Star Trek" prequel, Spock's father tells him, "You will always be a child of two worlds," urging him not to keep such a tight vise on his emotions. And Spandexy Old Spock, known as Spock Prime, tells his younger self: "Put aside logic. Do what feels right." – by Maureen Dowd, from her article in the New York Times on 10th May 2009.

We try to contemplate in this paper on how it would be intelligent when we say artificial intelligence. Human intelligence is not always efficient enough worthy to be called optimized but rather spontaneous or unpredictable more or less. Even when we come across a similar situation as before our behavior may be different than in the way we reacted then. Aiming such a flexibility in an artificial intelligent agent, we propose here a benchmark in which we have infinite number of equally valuable solutions. And then we observe if an agent trained by some of the machine learning techniques will behave in such an intelligent way as human's.

By *"path planing problem,"* we usually imply that our goal is to find a *shortest path* from a specified point to start to a specified point to finish. But what if our concern is to find the *longest path* to reach the goal with a limited amount of energy given? While a shortest-path-problem has a unique solution, this longest-path-problem has multiple solutions. Hence the latter will be a good benchmark to test if an agent can try a different route from one run to the next.

One option to realize this idea is that we make an agent learn during its action. The approach described here was motivated by the work by Floreano et al. (2000) in which the authors controlled mobile robots by a neural network so that the robots navigate properly by modifying synaptic weights of the neural network during navigation. The modification was based on a set of four Hebbian-like rules with each of the rules specified by a number of parameters. Each of the connection weights determines which rule with which parameters to modify itself during its navigation. Starting with a random configuration of the weights a population search eventually converges an optimized configuration. Later, Stanley (2003) united these four rules into one equation with two parameters. More recently, Durr (2008) proposed a more general equation of learning.

The experiments above were made using sigmoid neurons, that is, state takes a continuous value. We now are planning to use a neural network with spiking neurons. A counterpart of Hebbian learning is *spike-timing-dependent-plasticity*, or *STDP*.

A benchmark. To test how an agent behaves under some machine learning techniques, we propose here the following problem.

In a gridworld, agent moves to the neighboring cell spending one unit of energy. The gridworld is sufficiently large so that agent never reaches the border. Starting from the base located in the center of the gridworld with N units of energy, an agent must travel visiting as many different cells as possible, and the agent must go back to the base before consuming all the energy.

Try to imagine a land-rover in a planet like the Mars. The mission of the rover is to explore the areas which have not yet explored so far with a limited energy charged when it started the base.

Random walk as a basis of comparison. First of all, the rover explores with a random walk. The rover starts from the base, and moves up, down, left or right at random. The rover was given 100 units of fuel at the start. Since it's less likely to observe the rover will return to the base after a long journey, we observe if the rover tries to go as far as possible in the first half of the travel and tries to return to a point as close as possible to the base. To be more specific, what we want to know is, if it reached a point whose Manhattan distance from the base was more than M_{max}, while the final point it reached was the point of Manhattan distance less than M_{final} from the base. What we found then was, none out of 10,000,000 trials attained this target even for the condition of $M_{max} = M_{final} = 50$ which is a very weak demanding condition[1] and only one out of 100,000,000 trials fulfill this criteria, though it was not a good journey at all, that is, $M_{max} = 51$ and $M_{final} = 48$.

Reinforcement Learning. In a standard implementation of reinforcement learning, agent is given a reward only when it reaches to the goal. Hence, as Driessens et al. (2004) put it *"Using random exploration through the search space, rewards may simply never be encountered."* This might be a reason why usually those proposed world to be explored includes obstacles, and agent is given a penalty when it touches an obstacle. In a sense, obstacles might not be obstacle. Though the agent should avoid obstacles but at the same time those obstacles are good guides to lead the navigation. Our gridworld, however, has no border nor obstacle and includes only one goal to be reached, which makes the search *a-needle-in-a-haystack-problem*.

Our trick is, then, (i) agent is given a reward at any cell it passes; (ii) reward is associated with the Manhattan distance from the starting point; (iii) the further away the agent goes, the more reward agent obtains during its first half of the travel and vice versa during the last half of the travel.

Like the experiment with a random walk mentioned above, rover is given 100 units of fuel at the start. And what we observed was seven rovers after 300 episodes managed to return to the base after a journey. Further check, however, revealed the routes of all these seven rovers were just exploration close to the starting points. The main reason of this failure is *"the Q-function only converges after each state has been visited multiple times,"* as Driessens et al. (2004) pointed out.

Genetic Algorithm with a heuristic. We have found so far the problem is not so easy. However, we know a heuristic to create such a journey starting from the base with the goal being returning again to the base at the time when it has spent all the the fuels. That is, the number of up and down should be same, and same goes for the number of right and left. So, chromosome should be made up of four genes each corresponding to up, down, right, and left such that the number of these genes fulfill the condition above. It is not so difficult to create such chromosomes. For example, using a population of random permutations of the integer from 1 to N, the number of genes, we shuffle an original string which includes equal number of these four genes. Nevertheless, it is not so simple to apply genetic algorithm to this problem, because usual operation of both *crossover* and *mutation* destruct the condition, or validity of the journey.

So, we repair those broken chromosomes following Mitchell (2005) – one of the traditional approaches to the *Traveling Sales-person Problem*. The result was successful. See Figure 1.

Are we happy with this? Agents moves according to *chromosome* in genetic algorithm, and according to *policy* in reinforcement learning. As a result, the behavior is deterministic. Agent behaves any trials afterwards as exactly similar as when it learned. Can we call it a intelligence, however efficient the behavior might be? The answer would be "No," if our concern is on intelligence rather than efficiency.

A neural network approach toward an intelligent behavior. Then what we are planning is, to control an agent by spiking neurons with modifying its connection weights, from step to step, during navigation.

[1]We want the value to be more demanding ones, e.g., $M_{max} = 40$ and $M_{final} = 10$, but we found it was too demanding.

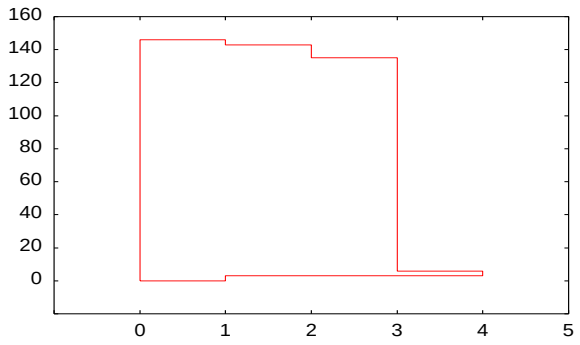

Figure 1: A successful journey from the base (0,0) to the base found by a genetic algorithm using a heuristic. The rover started the base with 300 units of fuel. The length of the loop was 300 cells.

Following the model by Florian (2005), we will exploit here neural networks of *stochastic leaky integrate-and-fire neurons*. Membrane potential $v_i(t)$ of neuron-i at time t evolves in discrete time δt according to:

$$v_i(t) = v_i(t - \delta t) \exp(-\delta t/\tau_i) + \sum_j w_{ij}(t - \delta t) f_j(t - \delta t)$$

where τ_i is a time constant of neuron-i, w_{ij} is synaptic weight value from neuron-j to neuron-i, and $f_j(t) = 1$ if neuron-j fires at time t otherwise 0.

The neuron-i fires stochastically with probability $\sigma(v_i(t))$

$$\sigma_i(v_i) = \begin{cases} \delta t/\tau_\sigma \exp(\beta_\sigma(v_i - \theta_i)) & \text{if the value is less than 1} \\ 1 & \text{otherwise} \end{cases}$$

If the neuron fires, the the membrane potential is reset to a reset potential V_r.

The parameters of neurons are set here as follows: $\delta t = 1$, a time constant $\tau_i = 20$ms, reset potential $V_r = 0$ mV, threshold potential $\theta_i = 5$ mV, exponential escape noise with $\tau_\sigma = 25$ ms and $\beta_\sigma = 0.2$ mV.

We experiment here, among others, with a feedforward architecture with two sensor neurons, input layer with 4 neurons, hidden layer with 8 neurons, output layer with 2 neurons. All neurons from one layer to the next layer are fully connected. At the beginning of a run, the synaptic weights were initialized with random values from -1 to 1 except for those from the sensor neurons which take from 0 to 1 at random.

Since we have no object, the activation of the sensor neurons takes a random value from between 0 and 1. The sensor neurons fired Poisson spike trains, proportional to the activation, with a firing rate 200 Hz.

The motor activations $a(t)$ of the output neurons evolve according to the following equation with time constant τ_e = 2s.

$$a(t) = a(t - \delta t) \exp(-\delta t/\tau_e) + (1 - \exp(-1/\nu_e \tau_e)) f(t).$$

The factor of $f(t)$ is to normalize the activation to 1 when the neuron fires regularly with frequency $\nu_e = 25$ Hz. Agent moves with its x and y coordinate being increment or decrement by these two motor neurons. (Note that the world is no more discrete gridworld.)

Thus we can make agent explore the world anyway. The question is, how we modify $w_{ij}(t)$ such that the agent solves the benchmark – navigate from the base to the base with maximum length of the path – at any ran with a different route. We hope we will be able to show some positive results in the conference.

Acknowledgment

This paper is based on the talk I made on 20th April 2009 in the seminar at the System Analysis Department of the Institute of Mathematics and Informatics, Vilnius, Lithuania. I thank for the stimulating and creative discussion at the seminar.

Reference

Floreano, D., and J. Urzelai (2000) "Evolutionary robots with online self-organization and behavioral fitness." Neural Networks Vol. 13, pp. 431–434.

Stanley, K. O., B. D. Bryant, and R. Miikkulainen (2003) "Evolving adaptive neural networks with and without adaptive synapses." Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2557–2564.

Durr, P., C. Mattiussi, A. Soltoggio and D. Floreano (2008) "Evolvability of neuromodulated learning for robots." Proceedings of ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems, pp. 41–46.

Driessens, K. and S. Dzeroski (2004) "Integrating guidance into relational reinforcement learning." Machine Learning, 57, pp.271-304.

Mitchell, G. G. (2005) "Validity constraints and the TSP - GeneRepair of genetic algorithms." Proceedings of Artificial Intelligence and Applications, pp. 306-311.

Florian, R. V. (2005) "A reinforcement learning algorithm for spiking neural networks." Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 299–306.