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.

Introduction. We contemplate in this paper on how it would be intelligent when we say *artificial intelligence*. Human intelligence is not always efficient, nor worthy enough 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 had 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 under a condition of a limited amount of energy given? While a shortest-path-problem has usually a unique solution, this longest-path-problem has multiple solutions. Therefore the latter might be a good benchmark to test if an agent can try a different route according to how the agent *feels* from one run to the next.

One option to realize this idea of flexible intelligence 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 a mobile robot by a neural network so that the robot navigates properly by modifying its 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 being specified by a number of parameters. Each connection determines which rule out of the four with which parameter values to modify itself during navigation. In other words, starting with a random configuration of the weights, each connection changes its weight value according to the rule assigned to the connection.

To obtain such a rule set, Floreano et al. exploited a learning via a population search in advance. The set of rules is evolved from generation to generation, and eventually those rules assigned to each of the connections converge to the optimal ones. This is called an *evolution of learning*.

Later, Stanley (2003) united these four rules into one equation with two parameters, with the meaning being remain intact. More recently, Durr (2008) proposed a more general equation of learning.[1]

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

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

[1]Floreano is one of the authors, too.

In a gridworld, agent moves to the neighboring cell spending one unit of energy. The gridworld is sufficiently large so that agents 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 return to the base before consuming all the energy given at its start.

Try to imagine an unmanned land-rover in a planet like the Mars. The mission of the rover is to explore the areas which have not yet explored with a limited fuels charged when it started the base. Of course the rover should return to the base before the tank of fuels becomes empty. To simply put, the rover should travel along a never-crossing-loop starting from the base and returning to the base again.

Random walk as a basis of comparison. First of all, let us make the rover explore 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 we found it was less likely to observe the rover would return to the base after a long journey, we observed if the rover tried to go as far as possible in the first half of the travel and tried 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 a preset value of M_{max}, while the final point it reached was the point of Manhattan distance less than M_{final}, the other preset value, from the base. What we found then was, none out of 10,000,000 trials attained this target even for a weak demanding condition of $M_{max} = M_{final} = 50$,[2] and only one out of 100,000,000 trials fulfilled 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 those proposed worlds to be explored usually include obstacles, and agent is given a penalty when it touches an obstacle. In a sense, obstacles might not be obstacle. Although the agent should avoid obstacles, those obstacles, on the other hand, 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 if it's during the first half of the travel, and vice versa if it's 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 explorations close to the starting points. The failure is because *"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 easy. However, we know a heuristic to create such a loop of journey starting from the base with the goal being returning again to the base at the time when it has spent all the fuels. That is, the number of up and down should be equal, 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 fulfills 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 those chromosomes, because a standard *crossover* as well as a standard *mutation* destruct the condition, or validity of the journey.

We repair those broken chromosomes so that they will recover its validity following Mitchell (2005) – one of the traditional approaches to the *Traveling Sales-person Problem.* The result was successful. See Figure 1.

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

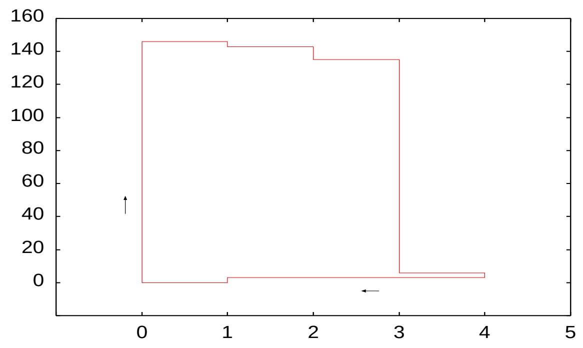

Figure 1: A successful journey from the base (0,0) to the base found by a genetic algorithm using a heuristic. The rover started the base with 300 units of fuel, and as such, the length of the loop was 300 cells.

Are we happy with this? In genetic algorithm, agents moves according to *chromosome*, and in reinforcement learning, moves are according to *policy*. As a result, the behavior is deterministic. Agent behaves any trials afterwards as exactly identical as when it learned. Can we call it an intelligence, however efficient the behavior might be? The answer would be "no," if our concern is on intelligence rather than on 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 one step to the next during a navigation.

Following the model by Florian (2005), we will use here a neural network 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: threshold potential $\theta_i = 5$ mV, exponential escape noise with $\tau_\sigma = 25$ ms, $\tau_i = 20$ms, $\delta t = 1$, $V_r = 0$ mV, 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 a value from 0 to 1 at random.

Since we have no obstacle, the activation of the sensor neurons takes a random value between 0 and 1. The sensor neurons fired Poisson spike trains, proportional to the activation, with a firing rate $r = 200$ Hz. Namely, the probability of emitting one spike during δt, is $r\delta t$.

The motor activations $a_i(t)$ ($i = 1, 2$ here) of the output neurons evolve according to the following equation with time constant $\tau_e = 2$s.

$$a_i(t) = a_i(t - \delta t)\exp(-\delta t/\tau_e) + (1 - \exp(-1/\nu_e \tau_e))f_i(t).$$

The factor of $f_i(t)$ is to normalize the activation to 1 when the neuron fires regularly with frequency $\nu_e = 25$ Hz. One output neuron's activity determines the distance r, the amount the agent moves at time t, and the other output neuron's activity determines the direction θ toward which the agent should move, that is, $\theta = 2\pi a_i(t)$ from the direction of the x-axis. Then agent moves with its increment being $\delta x = r\cos\theta$ and $\delta y = r\sin\theta$. Note that the world is no more discrete gridworld.

Thus we can make the agent explore the world anyway. See Figure 2. The question is, how we modify $w_{ij}(t)$ such that the point the agent finishes coincides to the point it started, with any ran being with a different route.

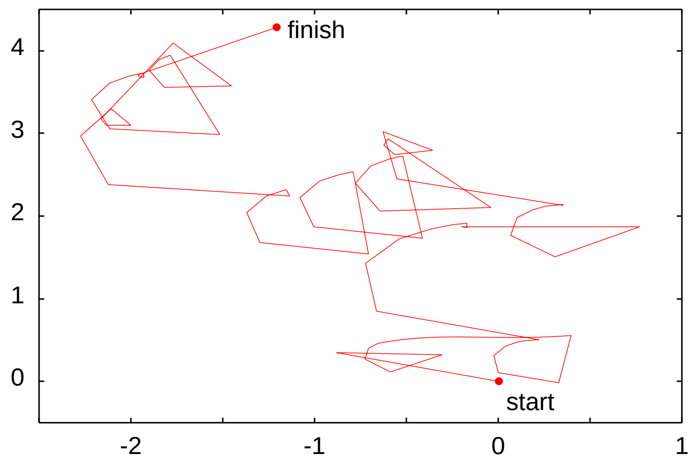

Figure 2: A route with 100 steps of the agent controlled by the feedforward spiking neuron's neural network described in the text.

A planned approach by STDP. One approach is by evolution. The principal equation of STDP is:

$$W(\Delta t) = \begin{cases} A_{(+)ij}\exp(-\Delta t/\tau_{(+)ij}) & \text{if} \quad \Delta t \geq 0 \\ -A_{(-)ij}\exp(-\Delta t/\tau_{(-)ij}) & \text{if} \quad \Delta t < 0 \end{cases}$$

where $\Delta t = t_{\text{post}} - t_{\text{pre}}$, difference of firing time of post-synaptic neuron and pre-synaptic neuron. Weight value changes depending on four parameters. Hence, each connection has to be assigned with these four parameters which will specify its weight value. Evolution is simply on those chromosomes which have $4N$ genes where N is the number of synaptic connections. Then question is, "what should be a fitness criteria?" Still we haven't solved this issue.

Yet another approach also by STDP. The other approach is by reinforcement learning, specifically by what Florian (2007) called *Reward Modulated STDP*. As for the learning of the synaptic weight values, Florian applies:

$$w_{ij}(t + \delta t) = w_{ij}(t) + \gamma r(t + \delta t)\zeta_{ij}(t)$$

where $r(t)$ is reward at time t and γ is *discount rate* by which eventual reward is estimated as

$$r(t) + \gamma r(t + \delta t) + \gamma^2 r(t + 2\delta t) + \gamma^3 r(t + 3\delta t) + \cdots.$$

Dynamics of ζ_{ij} is given by:

$$\zeta_{ij}(t) = P^{+}_{ij}(t)f_i(t) + P^{-}_{ij}(t)f_j(t),$$

in which $P_{ij}^{\pm}$ are:

$$P_{ij}^{+}(t) = P_{ij}^{+}(t - \delta t)\exp(-\delta t/\tau_{+}) + A_{+}f_j(t)$$

and

$$P_{ij}^{-}(t) = P_{ij}^{-}(t - \delta t)\exp(-\delta t/\tau_{-}) + A_{-}f_i(t)$$

where $\tau_{\pm}$ and $A_{\pm}$ are constant parameters.[3] See (Florian, 2007) for more in detail. Then the question is, "how should we set the reward $r(t)$?" We have not been successful, either.

Final Remarks. What we have described in this article is still an on-going project. The benchmark task proposed here was found to be difficult. Up to this moment, it has been resisted to be solved. It is this result that leads to the title of this paper: *"How can we design a more intelligent path-planning, if any?"* Neural networks by spiking neurons are said to be more biologically plausible. If so, the very simple benchmark proposed in this paper should be easily performed by exploiting spiking neurons, like our human brain does, but...

Acknowledgment

This paper is based on the talk the author made on the 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

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

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.

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

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.

Florian, R. V. (2007) "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity." Neural Computing, Vol. 19, No. 6. pp. 1468-1502.

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

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.

[3]For his benchmark of XOR, he set $\tau_{+} = \tau_{-} = 20ms$, $A_{+} = 1$, and $A_{-} = -1$. P_{ij}^{+} and P_{ij}^{-} track the influence of pre-synaptic and post-synaptic spikes, respectively (Florian 2007).