Direct Policy Computation by the Liouville Machine

Ivo Kwee and Jürgen Schmidhuber
IDSIA, Galleria 2, CH-6928 Manno, Switzerland.
E-mail: ivo@idsia.ch, juergen@idsia.ch

Abstract

We present a novel, continuous formulation of reinforcement learning naturally suited to problems involving realistic physics. Tasks are defined by physical boundary conditions. For instance, obstacles in a maze impose zero agent velocity, swamps reduced velocity; the agent's inertia and action set may restrict velocity changes. Given some task, we define a set of probability densities on continuous state and action and time. From these densities, the goal is to derive an optimal policy such that for all states the most likely action maximizes cumulative reward. Liouville's conservation theorem tells us that the density at time t, state s and action a, must equal the density at $t + dt$, $s + ds$, $a + da$. Discretization yields a coupled set of partial differential equations that can be solved directly and whose solution corresponds to an optimal policy. Discounted reward schemes are incorporated naturally by taking the Laplace transform of the equations. Experiments illustrate how the Liouville machine works.

1 Introduction

Traditional reinforcement learning (RL) algorithms (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996) derive from dynamic programming and the theory of discrete Markov Decision Processes (MDPs). For instance, there are convergence theorems (Kearns & Singh, 1999) for variants of Q-learning in finite state spaces (Watkins & Dayan, 1992).

Theoretical insights concerning RL in continuous worlds, however, are few. Typically heuristics are used to extend the discrete MDP framework to applications involving a continuous domain. For instance, nonlinear differentiable function approximators such as neural networks are used to map continuous (or huge) state spaces to predictions of cumulative future reward (Puterman, 1994; Bertsekas & Tsitsiklis, 1996). See Tesauro (1994) for a successful example.

Here we propose a radically different approach to RL whose natural setting is the continuous domain. It is conceptually much closer to physical transport equations than to the traditional MDP framework. RL in discrete domains is derived from the continuous variant, not the other way round as in the MDP approach.

The basic idea is to model the dynamics of an agent in some environment in analogy to the motion of scattering particles, and to interpret the well-known transport equation as a formalism for RL control tasks. The agent's learning task is to apply forces to the environment (to shape its probabilistic scattering behavior) such that a desired behavior is attained.

In what follows, we will first review Liouville's theorem which provides an essential constraint on the temporal evolution of the agent's most likely action. Then we will proceed to the transport equation taking into account possible collisions and absorption states (such as "death traps"). The transport equation naturally yields an algorithm optimizing the agent's behavior. Illustrative experiments will focus on agents with inertia solving maze tasks.

2 Derivation of the Transport Equation

We define the *action* as the unit vector in the direction of the velocity of an agent. Let $f(\boldsymbol{s}, \boldsymbol{a}, t)$ denote the joint probability density of finding the agent at position $\boldsymbol{s}$, executing action $\boldsymbol{a}$ at time t. We will establish a relation describing the evolution of this density in time, in analogy to the motion of particles described by transport theory.

Liouville's theorem

According to Liouville's theorem the (probability) density of a volume element is conserved along a flow line (Kittel & Kroemer, 1980), i.e.

$$f(\boldsymbol{s} + d\boldsymbol{s}, \boldsymbol{a} + d\boldsymbol{a}, t + dt) - f(\boldsymbol{s}, \boldsymbol{a}, t) = 0. \tag{1}$$

In other words, in absence of collisions or absorption states (e.g., death pits that eat agents), some agent's probability density must remain constant as we track its course in state-action space over a time step dt.

Eq. 1 is linearized by using partial derivatives with respect to all independent variables. This yields

$$\left[dt \frac{\partial}{\partial t} + d\boldsymbol{s} \cdot \nabla_s + d\boldsymbol{a} \cdot \nabla_a \right] f(\boldsymbol{s}, \boldsymbol{a}, t) = 0 \tag{2}$$

We divide both sides by dt to obtain

$$\left[\frac{\partial}{\partial t} + \boldsymbol{\alpha} \cdot \nabla_s + \boldsymbol{\beta} \cdot \nabla_a \right] f(\boldsymbol{s}, \boldsymbol{a}, t) = 0 \tag{3}$$

where ∇_s is the gradient operator in space and ∇_a is the gradient operator in action; $\boldsymbol{\alpha} = \partial \boldsymbol{s} / \partial t$ is a *velocity* vector (representing state change per time unit), and $\boldsymbol{\beta} = \partial \boldsymbol{a} / \partial t$ is an acceleration vector (representing velocity change per time unit). In transport theory, the sum of the operators in brackets is also called the *streaming operator*; the left hand side is called *substantial derivative.*

Transport equation

In what follows, the term "collision" will refer to *any* interaction of the agent with the environment, with other agents, or with itself. The term "absorption" will refer to the vanishing of an agent whose inertia and energy may be compeletely absorbed by a "death trap." The term "source" will refer to a point where an agent may enter the world.

Collisions require additional terms on the right hand side:

$$\left[\frac{\partial}{\partial t} + \boldsymbol{\alpha} \cdot \nabla_s + \boldsymbol{\beta} \cdot \nabla_a \right] f = \left(\frac{\partial f}{\partial t} \right)_{\text{coll}}. \tag{4}$$

It is the term $(\partial f / \partial t)_{\text{coll}}$ that accounts for density changes due to collisions, including source and absorption processes. The equation above is a generic form of the transport equation (Duderstadt & Martin, 1979), here applied to probability densities.[1] In general, parameters $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ are functions of $\boldsymbol{s}$, $\boldsymbol{a}$ and t.

[1] We may also add a source term to the right hand side. For example, in photon transport, the source will be a light source.

Particular forms of the collision term lead to alternative variants of the transport equation, such as the Boltzmann equation for gas dynamics, the neutron transport equation or the radiative transfer equation. In its most general form the collision term is

$$\left(\frac{\partial f}{\partial t}\right)_{\text{coll}} = \int_0^\infty \int_A \int_\Omega \Sigma(\boldsymbol{s}' \to \boldsymbol{s}, \boldsymbol{a}' \to \boldsymbol{a}, \boldsymbol{t} - \boldsymbol{t}')\, n(\boldsymbol{s}', \boldsymbol{a}', t')\, d\boldsymbol{r}' d\boldsymbol{a}' dt', \tag{5}$$

which defines *all* possible correlations in time, space and action. Hence the approach is not limited to Markovian environments where the current state in principle conveys all information about the probability of the next state, given a particular action — information about previous states is naturally taken into account if necessary.

3 Application to Reinforcement Learning

Interpretation of terms

We first take a closer look at the physical meaning of terms and parameters in the transport equation, then put the collision term into appropriate form. We start by rearranging the equation:

$$\frac{\partial f}{\partial t} = -\boldsymbol{\alpha} \cdot \nabla_s f - \boldsymbol{\beta} \cdot \nabla_a f + \left(\frac{\partial f}{\partial t}\right)_{\text{coll}}. \tag{6}$$

The first term on the right hand side describes local changes in the density caused by the agent's finite inertia and its *velocity*, $\boldsymbol{\alpha}$. The agent is arriving at or leaving a small volume element around $\boldsymbol{s}$.

The second term describes local density changes due to *"external"* forces changing the agent's direction. Recall that $\boldsymbol{\beta}$ describes an acceleration term, and that external forces on the agents are related to the latter by Newton's law: $\boldsymbol{\beta} = \boldsymbol{F}/m$, where m is the mass of the agent. If the force results from a potential field V, then $\boldsymbol{F} = -\nabla V$. Note that "external" does not necessarily mean "uncontrollable". External forces include gravity and those of "muscles" controlled by the agent itself.

The third term describes density changes due to collisions in the broad sense above. Absorption processes are also included in this term. This may sound rather strange, but in physical systems, absorption often occurs after a particle collides with another and all its energy is absorbed. The RL interpretation is an environmental state that may kill the agent with a certain probability.

To summarize, we model the dynamics of an agent in a continuous state space like the motion of scattering particles. The transport equation describes the evolution of the joint density of state-action pairs in time, and states that density changes occur because of agent movements due to external forces or collisions.

Exploration and non-Markovian environments

The collision term is associated with random density changes. We can determine the agent's "exploration behaviour" by defining the scattering kernel $\Sigma(\boldsymbol{s}' \to \boldsymbol{s}, \boldsymbol{a}' \to \boldsymbol{a}, t - t')$. For instanteneous scattering events, $\Sigma(\boldsymbol{s}' \to \boldsymbol{s}, \boldsymbol{a}' \to \boldsymbol{a}, t - t')\delta(t - t')\delta(\boldsymbol{s} - \boldsymbol{s}')$. For pure random exploration (both homogeneous and isotropic) the kernel is simply a constant, i.e. $\Sigma = c$; for time-invariant, isotropic but spatially dependent exploration we have $\Sigma = \Sigma(\boldsymbol{s})$.

The assumption of *instantaneous* scattering only permits correlations local in time. In most cases the latter also imply locality in space. This leads to a fundamental assumption of

traditional RL, namely, the *Markov assumption*. By relaxing the restriction of locality in time, however, we may also model *non-Markovian* environments that exhibit state changes depending on events earlier in time (compare Section 2). We then have to use the general form of the collision term in Eq. 5 that retains the convolution in time.

Discounted reward and Laplace transform

Traditional RL often considers discounted reward schemes. Consider a *time decaying* reward of unit amplitude of the form

$$\rho(\boldsymbol{s}, \boldsymbol{a}, t) = \delta(\boldsymbol{s} - \boldsymbol{s}^*, \boldsymbol{a} - \boldsymbol{a}^*)\, e^{-\gamma t} \tag{7}$$

where $\boldsymbol{s}^*$ and $\boldsymbol{a}^*$ denote the goal position and goal action, respectively. Then, the integrated reward is

$$R = \int_0^\infty \int_A \int_\Omega f(\boldsymbol{s}, \boldsymbol{a}, t)\, \rho(\boldsymbol{s}, \boldsymbol{a}, t)\, d\boldsymbol{s}\, d\boldsymbol{a}\, dt = F(\boldsymbol{s}^*, \boldsymbol{a}^*; \gamma) \tag{8}$$

where we have recognized the integral as a Laplace transform. The term $F(\boldsymbol{s}, \boldsymbol{a}, \gamma)$, on the right hand side, is related to $f(\boldsymbol{s}, \boldsymbol{a}, t)$ by the definition of the (one-sided) Laplace transform,

$$F(\boldsymbol{s}, \boldsymbol{a}; \gamma) = \mathcal{L}[f(\boldsymbol{s}, \boldsymbol{a}, t)] = \int_0^\infty f(\boldsymbol{s}, \boldsymbol{a}, t)\, e^{-\gamma t} dt. \tag{9}$$

We see that the discounted cumulative reward is directly related to the Laplace transform of our density $f(\boldsymbol{s}, \boldsymbol{a}, t)$. It is therefore convenient to consider the Laplace transform of the transport equation,

$$[\gamma + \boldsymbol{\alpha} \cdot \nabla_s + \boldsymbol{\beta} \cdot \nabla_a]\, F(\boldsymbol{s}, \boldsymbol{a}; \gamma) = C(\boldsymbol{s}, \boldsymbol{a}; \gamma). \tag{10}$$

On the right hand side, C is the Laplace transforms of the collision term. The reader might recognize that the Laplace decay parameter, γ, appears as a numerical absorption term in the equation. The solution of the Laplace-transformed transport equation represents a (exponentially) time-weighted probability of finding an agent in $(\boldsymbol{s}, \boldsymbol{a})$

4 Numerical Solution

The equations above are naturally formulated for continuous space, time and action. For computational solutions we need to discretize the transport equation (TE) and pose appropriate boundary conditions.

Simplifications and collision term

It is generally impossible to solve the TE directly because of its high dimensionality. Especially the collision term, involving multiple integrals and a convolution in time, can become quite complex. For sake of better calculability we start by making a few simplifying yet quite RL-typical assumptions.

1. Constant speed. We assume the speed of the agent to be constant and homogeneous.

2. Collisions occur instanteneously and with time-invariant probability. Roughly, this means that we assume the Markov property and that the scattering kernel is "memoryless".

3. Isotropic scattering. This means that the agent does not remember from which direction it came from, and scatters with equal probability in all directions (like undirected exploration in traditional RL).

Any of the assumptions above may be relaxed. This will require more computation, however, but add little in understanding the basic concepts of the Liouville Machine. Given the assumptions above, the collision term can be written as

$$\left(\frac{\partial f}{\partial t}\right)_{\text{coll}} = c\Sigma_0 \left[\frac{1}{2\pi}\int_A f(\boldsymbol{s}, \boldsymbol{a}', t)\, d\boldsymbol{a}' - f(\boldsymbol{s}, \boldsymbol{a}, t)\right], \tag{11}$$

where c denotes the velocity of the agent. In transport theory, Σ_0 is known as the scatter coefficient and represents the probability of a scatter event per distance travelled. In our application we may regard it just as a parameter that defines the amount of random exploration; if $\Sigma_0 = 0$ then there is no exploration at all.

We also assume $\boldsymbol{\beta} = \mathbf{0}$, i.e., we disregard any external forces on the agent. In the forthcoming maze example, this means that dynamics are controlled by the agents themselves.

Furthermore, we assume discounted reward with a *single* goal state of the form

$$\rho(\boldsymbol{s}, \boldsymbol{a}, t) = \delta(\boldsymbol{s}^* - \boldsymbol{s})\, \delta(\boldsymbol{a}^* - \boldsymbol{a}) e^{-\gamma t}. \tag{12}$$

Substitution into Eq. 10 yields a Laplace-transformed TE of the following form:

$$\mathcal{A}\left[F(\boldsymbol{s}, \boldsymbol{a}; \gamma)\right] = \; c\Sigma_0 \frac{1}{2\pi}\int_A F(\boldsymbol{s}, \boldsymbol{a}; \gamma)\, d\boldsymbol{a} \tag{13}$$

with the operator

$$\mathcal{A} = \gamma + \boldsymbol{\alpha} \cdot \nabla_s + c\Sigma_0. \tag{14}$$

Discretization of space and action in a maze

We use the discrete ordinates methods (Duderstadt & Martin, 1979) to discretize action space in four directions a_k, with $k = 1, ..., 4$, corresponding to *north*, *east*, *south* and *west*, respectively. The transport equation then reduces to a coupled set of four equations:

$$\mathcal{A}\left[F_k\right] = \; c\Sigma_0 \sum_{i=1}^{4} w_i\, p(\boldsymbol{a}_i \rightarrow \boldsymbol{a}_k)\, F_i, \qquad \text{for } k = 1, .., 4, \tag{15}$$

where $F_k = F(\boldsymbol{s}, \boldsymbol{a}_k; \gamma)$; weights w_i have to be correctly determined by quadrature.

For the spatial variable we use a finite difference scheme. We construct N mesh points in domain Ω and replace the gradient operators by their discrete counter parts. If we define $f_k^n = F(\boldsymbol{s}_n, \boldsymbol{a}_k; \gamma)$ and represent the density by a vector $\boldsymbol{f} = (f_1^1, f_1^1, ..., f_K^1, ..., f_K^N)$, the system of equations may be expressed in matrix form as follows:

$$\boldsymbol{A}\boldsymbol{f} = \boldsymbol{B}\boldsymbol{f} \tag{16}$$

where $\boldsymbol{A}$ represents the discretized operator of Eq. 14, and $\boldsymbol{B}$ is the discretized form of the integral transform on the right hand side of Eq. 13.

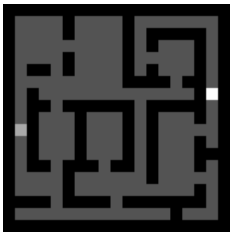

Figure 1: Layout of the example maze. The starting position is left. This maze has multiple solutions that lead to the goal position on the right.

Initial and boundary conditions

We still need to impose correct initial and boundary conditions before we can solve for $\boldsymbol{f}$. We start at t=0 with an agent at some initial state-action $(\boldsymbol{s}_0, \boldsymbol{a}_0)$, thus the initial condition is

$$f(\boldsymbol{s}, \boldsymbol{a}, 0) = F(\boldsymbol{s}, \boldsymbol{a}_k; \gamma) = \delta(\boldsymbol{s}_0 - \boldsymbol{s})\, \delta(\boldsymbol{a}_0 - \boldsymbol{a}). \tag{17}$$

For the boundary conditions, we require that

$$f(\boldsymbol{s}, \boldsymbol{a}, t) = F(\boldsymbol{s}, \boldsymbol{a}; \gamma) = 0 \tag{18}$$

whenever $\boldsymbol{s}$ is a wall. This boundary condition states that any agent that bumps into a wall is immediately removed. Alternatively one may model *reflective* boundaries, or require that the velocity becomes zero at the boundary. Here, however, our goal is just to illustrate and exemplify the basic scheme.

Solution method

Together with the initial and boundary condition, we obtain a linear system of the form

$$\boldsymbol{C}\boldsymbol{f} = \boldsymbol{b}, \tag{19}$$

where $\boldsymbol{b}$ is a vector containing the boundary values, and $\boldsymbol{C} = \boldsymbol{A} - \boldsymbol{B}$ with $\boldsymbol{A}$ and $\boldsymbol{B}$ as defined in Eq. 16. The solution can be found using conventional methods for solving linear systems. Because the matrix $\boldsymbol{C}$ is sparse but not symmetric, we have used a preconditioned biconjugate gradient sparse matrix solver.

5 Experiments

We report results of the Liouville machine for a maze with numerous obstacles. Figure 1 shows the layout of our test maze. The domain is discretized in a 19×19 square grid. In the following examples we have chosen $\Sigma_0 = 0.1$ correspondeding to an intermediate amount of exploration, and a small Laplace parameter of $\gamma = 0.01$ corresponding to a lightly discounted reward scheme.

The number of unknowns are $K \times N$ where K is the number of discretized directions and N is the dimensionality of the maze. In our case with $K = 4$ and $N = 19 \times 19$ we had a total number of 1444 unknowns. For this maze, computation times were typically less than 1 second on a Ultra-SPARC workstation.

Figure 2: Direct solutions of the maze. Left to right: probability density solutions for *north*, *east*, *south* and *west* directions, respectively. Rightmost plot: mean probability averaged over all directions.

Figure 3: Adjoint solutions of the maze. Left to right: probability density solutions for *north*, *east*, *south* and *west* directions, respectively. Rightmost plot: mean probability averaged over all directions.

Direct solution

Our first example is the solution of the direct problem: given the *starting state* of the agent, we determine the agent's probability density in the entire maze. The probability of the agent arriving at the goal state is just the density value of the solution at the goal.

Figure 2 shows the solutions. Each graph plots the distribution of the probability density for the directions *north*, *east*, *south* and *west*, respectively. Notice the structural differences between the solutions for the different directions due to the agent's inertia. The rightmost graph plots the average density over all directions and corresponds to the probability of finding the agent at a position irrespective of its current action.

Adjoint solution

In the second example we solve the so-called *adjoint problem*: given that the agent arrives at the *goal state*, we solve for the probability density of the agent coming from other states. Interestingly, the solution is obtained by specifying the goal state as initial condition, and solving the transport equation using "negative velocity".

Figure 3 shows the adjoint solution. Again each plot corresponds to the probability density for the *north*, *east*, *south* and *west* directions, respectively; the rightmost plot is the average density.

Path conditional density

We can now also solve for the (path) conditional probability of an agent going through an intermediate state $C = (\boldsymbol{s}, \boldsymbol{a})$, that has started from $A = (\boldsymbol{s}_0, \boldsymbol{a}_0)$ and arrives at $B = (\boldsymbol{s}^*, \boldsymbol{a}^*)$. Using Bayes' formula we have

$$p(C|A,B) \propto p(B|C)\, p(C|A), \tag{20}$$

Figure 4: Path conditional solutions for the maze. Left to right: probability density solutions for *north*, *east*, *south* and *west* directions, respectively. Rightmost plot: mean probability averaged over all directions.

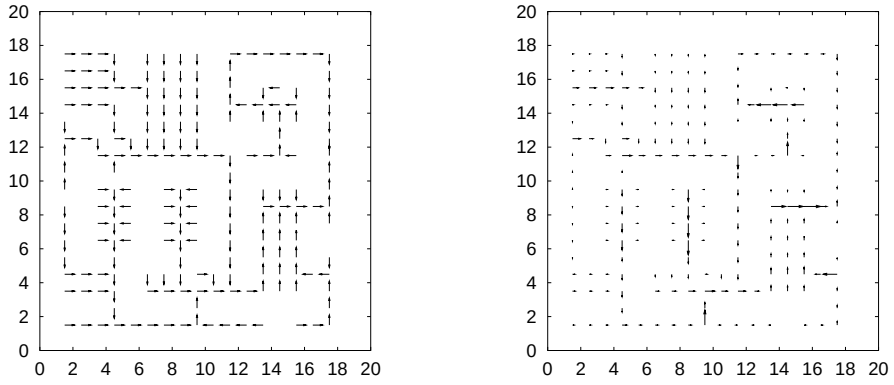

Figure 5: Optimal policy of the maze. (a) Normalized plot of maximum probable direction. (b) "advantage"-weighted plot of maximum probable direction.

where we have used $p(B|C) = p(B|C, A)$ because the probability of reaching the goal if conditioned on the intermediate state C is independent of its initial state A. Thus the conditional density $p(C|A, B)$ is simply obtained by multiplication of the *direct* map $p(C|A)$ and *adjoint* map $p(B|C)$, which both have been computed earlier.

Figure 4 shows solutions of this path conditional density. Notice the two "rooms" at the approximate center of the maze that remain dark, because the low probability of an agent passing these areas. The path conditional density represents the (pointwise) probability of passing a state of an agent that comes from the starting state and is known to have reached the goal. So we obtain a way of dealing with subgoals as a natural byproduct of the Liouville machine.

Optimal policy

The optimal policy is obtained by choosing the most probable action (i.e. the direction which corresponds to the highest probability) from the adjoint map. The optimal policy is derived from the adjoint solution, rather than from the path conditional density, because the policy should be independent from the starting position.

Plot (a) in Figure 5 shows the optimal policy for our maze. However, at some states the optimal action is actually only marginally better than the other directions. To show this, we have drawn the same graph again but weighted the arrows by their *advantage*-value which was defined as the difference between the optimal value and the mean value (see Plot (b) in the same figure).

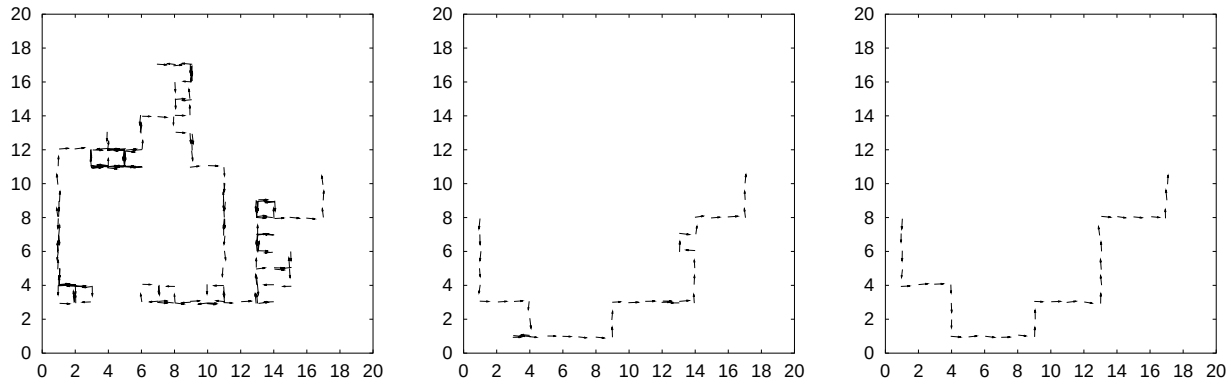

Figure 6: Generated paths computed from adjoint map. (a) Random path for $\gamma = 0.01$ (202 steps). (b) Random path for $\gamma = 0.1$ (42 steps). (c) Shortest path (34 steps). All paths were generated from the adjoint density solution and required typically less than 1 second on a Ultra-SPARC workstation.

Optimal path

From the probability density solution we can simulate the dynamic behaviour of agents in the maze. Plot (a) in Figure 6 shows a random path for a Laplace parameter $\gamma = 0.01$ that was generated by sampling directions according to the probability density solution; the length of the trail was 202 steps. We can force the agent to be more efficient by increasing the Laplace parameter to $\gamma = 0.1$, which corresponds to increased discount of the reward; Plot (b) in the same figure shows the resulting generated trail. Notice that the agent "switches" to a different solution that now goes through the lower part of the maze; this policy is almost optimal and it takes 42 steps to reach the goal. Finally, the optimal path can be determined by taking the *most* probable action in each state. Plot (c) shows the shortest path through the maze which required a total of 34 steps from the start to the goal.

At first sight, it may seem unclear how the Liouville machine has obtained the optimal policy from the probability distributions from a random walk. It may seem contradictive to have formulated the explorative behaviour in Eq. 11 as isotropic in all directions, while the obtained optimal path in Fig. 6 surely is not. To understand, we must remember that the probability density solution is a cumulation of *all* possible paths; a cumulation that includes both optimal and non-optimal solutions. However, early arriving agents *must* have travelled along short-path solutions, and the Laplace transform acts as an artificial time-weighted filter that is able to extract these "fast" agents. The optimal path can be obtained either in the limit $\gamma \to \infty$, or, as previously shown, just by selecting the most probable action at each state.

6 Conclusion

We have derived a transport equation for the time-dependent joint probability of state-action pairs using Liouville's theorem of conservation. This yields a natural formulation of a wide variety of reinforcement learning (RL) problems involving continuous states and time, and may represent the birth of a general theory of continuous RL.

In ongoing theoretical work we are trying to establish a closer link between Liouville machine and more traditional RL approaches. Our formalism directly solves for the joint probability of state-action pairs. How exactly does this relate to value function and Q-values in discrete

RL based on MDP theory? In ongoing experimental work we are also applying the Liouville machine to more and more complex problems involving realistic physics and agent dynamics.

Acknowledgements

This work was sponsored by SNF grant 21-55409.98.

References

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming.* Athena Scientific, Belmont, MA.

Duderstadt, J. J., & Martin, W. R. (1979). *Transport theory.* John Wiley & Sons, New York.

Kearns, M., & Singh, S. (1999). Finite-sample convergence rates for Q-learning and indirect algorithms. *Advances in Neural Information Processing Systems 12.* MIT Press, Cambridge MA.

Kittel, C., & Kroemer, H. (1980). *Thermal physics.* W. H. Freeman and Co., New York.

Puterman, M. L. (1994). *Markov decision processes.* John Wiley & Sons, New York.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning. An introduction.* MIT Press, Cambridge.

Tesauro, G. (1994). TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, *6*, 215–219.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.